

# **EXHIBIT 1**

## Exhibit Copyright-A

### High Level Comparison of Java and Android API Specs

#### Java™ 2 Platform Standard Edition 5.0 API Specification

**Java™ 2 Platform Standard Edition 5.0 API Specification**

This document is the API specification for the Java 2 Platform Standard Edition 5.0.

See: [Description](#)

Package	Description
<a href="#">java.applet</a>	Provides the classes necessary to create an applet and the classes an applet uses to communicate with its applet context.
<a href="#">java.awt</a>	Contains all of the classes for creating user interfaces and for painting graphics and images.
<a href="#">java.awt.color</a>	Provides classes for color spaces.
<a href="#">java.awt.datatransfer</a>	Provides interfaces and classes for transferring data between and within applications.
<a href="#">java.awt.dnd</a>	Drag and Drop is a direct manipulation gesture found in many Graphical User Interface systems that provides a mechanism to transfer information between two entities logically associated with presentation elements in the GUI.
<a href="#">java.awt.event</a>	Provides interfaces and classes for dealing with different types of events fired by AWT components.
<a href="#">java.awt.font</a>	Provides classes and interface relating to fonts.
<a href="#">java.awt.geom</a>	Provides the Java 2D classes for defining and performing operations on objects related to two-dimensional geometry.
<a href="#">java.awt.im</a>	Provides classes and interfaces for the input method framework.

#### Android APIs

**Package Index**

These are the Android APIs.

<a href="#">android</a>	Contains the resource classes used by standard Android applications.
<a href="#">android.accessibilityservice</a>	
<a href="#">android.accounts</a>	
<a href="#">android.app</a>	High-level classes encapsulating the overall Android application model.
<a href="#">android.app.admin</a>	
<a href="#">android.app.backup</a>	Contains the backup and restore functionality available to applications. If a user wipes the data on their device or upgrades to a new Android-powered device, all applications that have enabled backup will restore the user's previous data.  For a detailed guide to using the backup APIs, see the <a href="#">Data Backup developer guide</a> .

## Java™ 2 Platform Standard Edition 5.0 API Specification

The screenshot shows the Java 2 Platform Standard Edition 5.0 API Specification page for the `java.util` package. The page is titled "Package java.util" and contains a description: "Contains the collections framework, legacy collection classes, event model, date and time facilities, internationalization, and miscellaneous utility classes (a string tokenizer, a random-number generator, and a bit array)." It also includes a "See:" section with a link to "Description". Below this is an "Interface Summary" table listing various interfaces and their descriptions.

Interface	Description
<a href="#">Collection&lt;E&gt;</a>	The root interface in the <i>collection hierarchy</i> .
<a href="#">Comparator&lt;T&gt;</a>	A comparison function, which imposes a <i>total ordering</i> on some collection of objects.
<a href="#">Enumeration&lt;E&gt;</a>	An object that implements the <i>Enumeration</i> interface generates a series of elements, one at a time.
<a href="#">EventListener</a>	A tagging interface that all event listener interfaces must extend.
<a href="#">Formattable</a>	The <i>Formattable</i> interface must be implemented by any class that needs to perform custom formatting using the 's' conversion specifier of <i>Formatter</i> .
<a href="#">Iterator&lt;E&gt;</a>	An iterator over a collection.
<a href="#">List&lt;E&gt;</a>	An ordered collection (also known as a <i>sequence</i> ).
<a href="#">ListIterator&lt;E&gt;</a>	An iterator for lists that allows the programmer to traverse the list in either direction, modify the list during iteration, and obtain the iterator's current position in the list.
<a href="#">Map&lt;K,V&gt;</a>	An object that maps keys to values.
<a href="#">Map.Entry&lt;K,V&gt;</a>	A map entry (key-value pair).
<a href="#">Observer</a>	A class can implement the <i>Observer</i> interface when it wants to be informed of changes in observable objects.

## Android APIs

The screenshot shows the Android Developers website for the `java.util` package. The page is titled "package java.util" and includes a description: "Provides an extensive set of utility classes." It also features a "more..." link and a table of interfaces.

Interface	Description
<a href="#">Collection&lt;E&gt;</a>	<i>Collection</i> is the root of the collection hierarchy.
<a href="#">Comparator&lt;T&gt;</a>	A <i>Comparator</i> is used to compare two objects to determine their ordering with respect to each other.
<a href="#">Enumeration&lt;E&gt;</a>	A legacy iteration interface.
<a href="#">EventListener</a>	<i>EventListener</i> is the superclass of all event listener interfaces.
<a href="#">Formattable</a>	Classes that handle custom formatting for the 's' specifier of <i>Formatter</i> should implement the <i>Formattable</i> interface.

## Java™ 2 Platform Standard Edition 5.0 API Specification

**Package java.beans**

Contains classes related to developing *beans* -- components based on the JavaBeans™ architecture.

See: [Description](#)

Interface Summary	
<a href="#">AppletInitializer</a>	This interface is designed to work in collusion with java.beans Beans instantiate.
<a href="#">BeanInfo</a>	A bean implementor who wishes to provide explicit information about their bean may provide a BeanInfo class that implements this BeanInfo interface and provides explicit information about the methods, properties, events, etc. of their bean.
<a href="#">Customizer</a>	A customizer class provides a complete custom GUI for customizing a target Java Bean.
<a href="#">DesignMode</a>	This interface is intended to be implemented by, or delegated from, instances of java.beans beancontext.BeanContext, in order to propagate to its nested hierarchy of java.beans beancontext.BeanContextChild instances, the current "designTime" property.
<a href="#">ExceptionListener</a>	An ExceptionListener is notified of internal exceptions.
<a href="#">PropertyChangeListener</a>	A "PropertyChange" event gets fired whenever a bean changes a "bound" property.
<a href="#">PropertyEditor</a>	A PropertyEditor class provides support for GUIs that want to allow users to edit a property value of a given type.
<a href="#">VetoableChangeListener</a>	A VetoableChange event gets fired whenever a bean changes a "constrained" property.
<a href="#">Visibility</a>	Under some circumstances a bean may be run on servers where a GUI is not available.

## Android APIs

**package java.beans**

Since: API Level 3

**Interfaces**

- [PropertyChangeListener](#) A PropertyChangeListener can subscribe with a event source.

**Classes**

<a href="#">IndexedPropertyChangeEvent</a>	A type of <a href="#">PropertyChangeEvent</a> that indicates that an indexed property has changed.
<a href="#">PropertyChangeEvent</a>	An event that indicates that a constraint or a boundary of a property has changed.
<a href="#">PropertyChangeListenerProxy</a>	The implementation of this listener proxy just delegates the received events to its listener.
<a href="#">PropertyChangeSupport</a>	This utility class

## Exhibit Copyright-B

Java™ 2 Platform Standard Edition 5.0 API Specification (java.lang)		Android APIs (java.lang)	
<b>Interface Summary</b>		<b>Interfaces</b>	
<a href="#"><u>Appendable</u></a>	An object to which <code>char</code> sequences and values can be appended.	<a href="#"><u>Appendable</u></a>	Declares methods to append characters or character sequences.
<a href="#"><u>CharSequence</u></a>	A <code>CharSequence</code> is a readable sequence of <code>char</code> values.	<a href="#"><u>CharSequence</u></a>	This interface represents an ordered set of characters and defines the methods to probe them.
<a href="#"><u>Cloneable</u></a>	A class implements the <code>Cloneable</code> interface to indicate to the <a href="#"><u>Object.clone()</u></a> method that it is legal for that method to make a field-for-field copy of instances of that class.	<a href="#"><u>Cloneable</u></a>	This (empty) interface must be implemented by all classes that wish to support cloning.
<a href="#"><u>Comparable&lt;T&gt;</u></a>	This interface imposes a total ordering on the objects of each class that implements it.	<a href="#"><u>Comparable&lt;T&gt;</u></a>	This interface should be implemented by all classes that wish to define a <i>natural order</i> of their instances.
<a href="#"><u>Iterable&lt;T&gt;</u></a>	Implementing this interface allows an object to be the target of the "foreach" statement.	<a href="#"><u>Iterable&lt;T&gt;</u></a>	Instances of classes that implement this interface can be used with the enhanced for loop.
<a href="#"><u>Readable</u></a>	A <code>Readable</code> is a source of characters.	<a href="#"><u>Readable</u></a>	Represents a sequence of characters that can be incrementally read (copied) into a <a href="#"><u>CharBuffer</u></a> .
<a href="#"><u>Runnable</u></a>	The <code>Runnable</code> interface should be implemented by any class whose instances are intended to be executed by a thread.	<a href="#"><u>Runnable</u></a>	Represents a command that can be executed.
<a href="#"><u>Thread.UncaughtExceptionHandler</u></a>	Interface for handlers invoked when a <code>Thread</code> abruptly terminates due to an uncaught exception.	<a href="#"><u>Thread.UncaughtExceptionHandler</u></a>	Implemented by objects that want to handle cases where a thread is being terminated by an uncaught exception.
<b>Class Summary</b>		<b>Classes</b>	

Java™ 2 Platform Standard Edition 5.0 API Specification (java.lang)		Android APIs (java.lang)	
<a href="#"><u>Boolean</u></a>	The Boolean class wraps a value of the primitive type <code>boolean</code> in an object.	<a href="#"><u>Boolean</u></a>	The wrapper for the primitive type <code>boolean</code> .
<a href="#"><u>Byte</u></a>	The Byte class wraps a value of primitive type <code>byte</code> in an object.	<a href="#"><u>Byte</u></a>	The wrapper for the primitive type <code>byte</code> .
<a href="#"><u>Character</u></a>	The Character class wraps a value of the primitive type <code>char</code> in an object.	<a href="#"><u>Character</u></a>	The wrapper for the primitive type <code>char</code> .
<a href="#"><u>Character.Subset</u></a>	Instances of this class represent particular subsets of the Unicode character set.	<a href="#"><u>Character.Subset</u></a>	
<a href="#"><u>Character.Unicode Block</u></a>	A family of character subsets representing the character blocks in the Unicode specification.	<a href="#"><u>Character.Unicode Block</u></a>	Represents a block of Unicode characters, as defined by the Unicode 4.0.1 specification.
<a href="#"><u>Class&lt;T&gt;</u></a>	Instances of the class <code>Class</code> represent classes and interfaces in a running Java application.	<a href="#"><u>Class&lt;T&gt;</u></a>	The in-memory representation of a Java class.
<a href="#"><u>ClassLoader</u></a>	A class loader is an object that is responsible for loading classes.	<a href="#"><u>ClassLoader</u></a>	Loads classes and resources from a repository.
<a href="#"><u>Compiler</u></a>	The <code>Compiler</code> class is provided to support Java-to-native-code compilers and related services.	<a href="#"><u>Compiler</u></a>	Placeholder class for environments which explicitly manage the action of a <i>Just In Time (JIT)</i> compiler.
<a href="#"><u>Double</u></a>	The Double class wraps a value of the primitive type <code>double</code> in an object.	<a href="#"><u>Double</u></a>	The wrapper for the primitive type <code>double</code> .
<a href="#"><u>Enum&lt;E extends Enum&lt;E&gt;&gt;</u></a>	This is the common base class of all Java language enumeration types.	<a href="#"><u>Enum&lt;E extends Enum&lt;E&gt;&gt;</u></a>	The superclass of all enumerated types.
<a href="#"><u>Float</u></a>	The Float class wraps a value of primitive type <code>float</code> in an object.	<a href="#"><u>Float</u></a>	The wrapper for the primitive type <code>float</code> .
<a href="#"><u>InheritableThreadLocal&lt;T&gt;</u></a>	This class extends <code>ThreadLocal</code> to provide inheritance of values from parent thread to child thread: when a child thread is created, the child receives initial values for all inheritable thread-local variables for which the parent has values.	<a href="#"><u>InheritableThreadLocal&lt;T&gt;</u></a>	A thread-local variable whose value is passed from parent to child thread.

Java™ 2 Platform Standard Edition 5.0 API Specification (java.lang)		Android APIs (java.lang)	
<a href="#"><u>Integer</u></a>	The Integer class wraps a value of the primitive type <code>int</code> in an object.	<a href="#"><u>integer</u></a>	The wrapper for the primitive type <code>int</code> .
<a href="#"><u>Long</u></a>	The Long class wraps a value of the primitive type <code>long</code> in an object.	<a href="#"><u>Long</u></a>	The wrapper for the primitive type <code>long</code> .
<a href="#"><u>Math</u></a>	The class <code>Math</code> contains methods for performing basic numeric operations such as the elementary exponential, logarithm, square root, and trigonometric functions.	<a href="#"><u>Math</u></a>	Class <code>Math</code> provides basic math constants and operations such as trigonometric functions, hyperbolic functions, exponential, logarithms, etc.
<a href="#"><u>Number</u></a>	The abstract class <code>Number</code> is the superclass of classes <code>BigDecimal</code> , <code>BigInteger</code> , <code>Byte</code> , <code>Double</code> , <code>Float</code> , <code>Integer</code> , <code>Long</code> , and <code>Short</code> .	<a href="#"><u>Number</u></a>	The abstract superclass of the classes which represent numeric base types (that is <code>Byte</code> , <code>Short</code> , <code>Integer</code> , <code>Long</code> , <code>Float</code> , and <code>Double</code> ).
<a href="#"><u>Object</u></a>	Class <code>Object</code> is the root of the class hierarchy.	<a href="#"><u>Object</u></a>	The root class of the Java class hierarchy.
<a href="#"><u>Package</u></a>	<code>Package</code> objects contain version information about the implementation and specification of a Java package.	<a href="#"><u>Package</u></a>	Contains information about a Java package.
<a href="#"><u>Process</u></a>	The <a href="#"><u>ProcessBuilder.start()</u></a> and <a href="#"><u>Runtime.exec</u></a> methods create a native process and return an instance of a subclass of <code>Process</code> that can be used to control the process and obtain information about it.	<a href="#"><u>Process</u></a>	Represents an external process.
<a href="#"><u>ProcessBuilder</u></a>	This class is used to create operating system processes.	<a href="#"><u>ProcessBuilder</u></a>	Creates operating system processes.
<a href="#"><u>Runtime</u></a>	Every Java application has a single instance of class <code>Runtime</code> that allows the application to interface with the environment in which the application is running.	<a href="#"><u>Runtime</u></a>	Allows Java applications to interface with the environment in which they are running.
<a href="#"><u>RuntimePermission</u></a>	This class is for runtime permissions.	<a href="#"><u>RuntimePermission</u></a>	Represents the permission to execute a runtime-related function.
<a href="#"><u>SecurityManager</u></a>	The security manager is a class that allows		

Java™ 2 Platform Standard Edition 5.0 API Specification (java.lang)		Android APIs (java.lang)	
	applications to implement a security policy.	<a href="#">SecurityManager</a>	<b>Warning:</b> security managers do <b>not</b> provide a secure environment for executing untrusted code.
<a href="#">Short</a>	The <code>Short</code> class wraps a value of primitive type <code>short</code> in an object.	<a href="#">Short</a>	The wrapper for the primitive type <code>short</code> .
<a href="#">StackTraceElement</a>	An element in a stack trace, as returned by <a href="#">Throwable.getStackTrace()</a> .	<a href="#">StackTraceElement</a>	A representation of a single stack frame.
<a href="#">StrictMath</a>	The class <code>StrictMath</code> contains methods for performing basic numeric operations such as the elementary exponential, logarithm, square root, and trigonometric functions.	<a href="#">StrictMath</a>	Class <code>StrictMath</code> provides basic math constants and operations such as trigonometric functions, hyperbolic functions, exponential, logarithms, etc.
<a href="#">String</a>	The <code>String</code> class represents character strings.	<a href="#">String</a>	An immutable sequence of characters/code units ( <code>chars</code> ).
<a href="#">StringBuffer</a>	A thread-safe, mutable sequence of characters.	<a href="#">StringBuffer</a>	A modifiable <a href="#">sequence of characters</a> for use in creating strings, where all accesses are synchronized.
<a href="#">StringBuilder</a>	A mutable sequence of characters.	<a href="#">StringBuilder</a>	A modifiable <a href="#">sequence of characters</a> for use in creating strings.
<a href="#">System</a>	The <code>System</code> class contains several useful class fields and methods.	<a href="#">System</a>	Provides access to system-related information and resources including standard input and output.
<a href="#">Thread</a>	A <i>thread</i> is a thread of execution in a program.	<a href="#">Thread</a>	A <a href="#">Thread</a> is a concurrent unit of execution.
<a href="#">ThreadGroup</a>	A thread group represents a set of threads.	<a href="#">ThreadGroup</a>	<a href="#">ThreadGroup</a> is a means of organizing threads into a hierarchical structure.
<a href="#">ThreadLocal&lt;T&gt;</a>	This class provides thread-local variables.	<a href="#">ThreadLocal&lt;T&gt;</a>	implements a thread-local storage, that is, a variable for which each thread has its own value.
<a href="#">Throwable</a>	The <code>Throwable</code> class is the superclass of all errors and exceptions in the Java language.	<a href="#">Throwable</a>	The superclass of all classes which can be thrown by the virtual machine.
<a href="#">Void</a>	The <code>Void</code> class is an uninstantiable placeholder		



Java™ 2 Platform Standard Edition 5.0 API Specification (java.lang)		Android APIs (java.lang)	
	class to hold a reference to the Class object representing the Java keyword void.	Void	Placeholder class for the Java keyword <code>void</code> .
<b>Enum Summary</b>		<b>Enums</b>	
<a href="#"><u>Thread.State</u></a>	A thread state.	Thread.State	A representation of a thread's state.
<b>Exception Summary</b>		<b>Exceptions</b>	
<a href="#"><u>ArithmeticException</u></a>	Thrown when an exceptional arithmetic condition has occurred.	ArithmeticException	Thrown when the an invalid arithmetic operation is attempted.
<a href="#"><u>ArrayIndexOutOfBoundsException</u></a>	Thrown to indicate that an array has been accessed with an illegal index.	ArrayIndexOutOfBoundsException	Thrown when the an array is indexed with a value less than zero, or greater than or equal to the size of the array.
<a href="#"><u>ArrayStoreException</u></a>	Thrown to indicate that an attempt has been made to store the wrong type of object into an array of objects.	ArrayStoreException	Thrown when a program attempts to store an element of an incompatible type in an array.
<a href="#"><u>ClassCastException</u></a>	Thrown to indicate that the code has attempted to cast an object to a subclass of which it is not an instance.	ClassCastException	Thrown when a program attempts to cast a an object to a type with which it is not compatible.
<a href="#"><u>ClassNotFoundException</u></a>	Thrown when an application tries to load in a class through its string name using: The <code>forName</code> method in class <code>Class</code> .	ClassNotFoundException	Thrown when a class loader is unable to find a class.
<a href="#"><u>CloneNotSupportedException</u></a>	Thrown to indicate that the <code>clone</code> method in class <code>Object</code> has been called to clone an object, but that the object's class does not implement the <code>Cloneable</code> interface.	CloneNotSupportedException	Thrown when a program attempts to clone an object which does not support the <code>Cloneable</code> interface.
<a href="#"><u>EnumConstantNotPresentException</u></a>	Thrown when an application tries to access an enum constant by name and the enum type contains no constant with the specified name.	EnumConstantNotPresentException	Thrown if an <code>enum</code> constant does not exist for a particular name.

Java™ 2 Platform Standard Edition 5.0 API Specification (java.lang)		Android APIs (java.lang)	
<a href="#"><u>Exception</u></a>	The class <code>Exception</code> and its subclasses are a form of <code>Throwable</code> that indicates conditions that a reasonable application might want to catch.	<a href="#"><u>Exception</u></a>	<code>Exception</code> is the superclass of all classes that represent recoverable exceptions.
<a href="#"><u>IllegalAccess Exception</u></a>	An <code>IllegalAccessException</code> is thrown when an application tries to reflectively create an instance (other than an array), set or get a field, or invoke a method, but the currently executing method does not have access to the definition of the specified class, field, method or constructor.	<a href="#"><u>IllegalAccess Exception</u></a>	Thrown when a program attempts to access a field or method which is not accessible from the location where the reference is made.
<a href="#"><u>IllegalArgumentEx ception</u></a>	Thrown to indicate that a method has been passed an illegal or inappropriate argument.	<a href="#"><u>IllegalArgument Exception</u></a>	Thrown when a method is invoked with an argument which it can not reasonably deal with.
<a href="#"><u>IllegalMonitor StateException</u></a>	Thrown to indicate that a thread has attempted to wait on an object's monitor or to notify other threads waiting on an object's monitor without owning the specified monitor.	<a href="#"><u>IllegalMonitorState Exception</u></a>	Thrown when a monitor operation is attempted when the monitor is not in the correct state, for example when a thread attempts to exit a monitor which it does not own.
<a href="#"><u>IllegalState Exception</u></a>	Signals that a method has been invoked at an illegal or inappropriate time.	<a href="#"><u>IllegalState Exception</u></a>	Thrown when an action is attempted at a time when the virtual machine is not in the correct state.
<a href="#"><u>IllegalThread StateException</u></a>	Thrown to indicate that a thread is not in an appropriate state for the requested operation.	<a href="#"><u>IllegalThreadState Exception</u></a>	Thrown when an operation is attempted which is not possible given the state that the executing thread is in.
<a href="#"><u>IndexOutOf BoundsException</u></a>	Thrown to indicate that an index of some sort (such as to an array, to a string, or to a vector) is out of range.	<a href="#"><u>IndexOutOfBounds Exception</u></a>	Thrown when a program attempts to access a value in an indexable collection using a value which is outside of the range of valid indices.
<a href="#"><u>Instantiation Exception</u></a>	Thrown when an application tries to create an instance of a class using the <code>newInstance</code> method in class <code>Class</code> , but the specified class object cannot be instantiated because it is an	<a href="#"><u>Instantiation Exception</u></a>	Thrown when a program attempts to access a constructor which is not accessible from the location where the reference is made.

Java™ 2 Platform Standard Edition 5.0 API Specification (java.lang)		Android APIs (java.lang)	
	interface or is an abstract class.		
<a href="#"><u>InterruptedException</u></a>	Thrown when a thread is waiting, sleeping, or otherwise paused for a long time and another thread interrupts it using the <code>interrupt</code> method in class <code>Thread</code> .	<a href="#"><u>InterruptedException</u></a>	Thrown when a waiting thread is activated before the condition it was waiting for has been satisfied.
<a href="#"><u>NegativeArraySizeException</u></a>	Thrown if an application tries to create an array with negative size.	<a href="#"><u>NegativeArraySizeException</u></a>	Thrown when an attempt is made to create an array with a size of less than zero.
<a href="#"><u>NoSuchFieldException</u></a>	Signals that the class doesn't have a field of a specified name.	<a href="#"><u>NoSuchFieldException</u></a>	Thrown when the virtual machine notices that a program tries to reference, on a class or object, a field that does not exist.
<a href="#"><u>NoSuchMethodException</u></a>	Thrown when a particular method cannot be found.	<a href="#"><u>NoSuchMethodException</u></a>	Thrown when the virtual machine notices that a program tries to reference, on a class or object, a method that does not exist.
<a href="#"><u>NullPointerException</u></a>	Thrown when an application attempts to use <code>null</code> in a case where an object is required.	<a href="#"><u>NullPointerException</u></a>	Thrown when a program tries to access a field or method of an object or an element of an array when there is no instance or array to use, that is if the object or array points to <code>null</code> .
<a href="#"><u>NumberFormatException</u></a>	Thrown to indicate that the application has attempted to convert a string to one of the numeric types, but that the string does not have the appropriate format.	<a href="#"><u>NumberFormatException</u></a>	Thrown when an invalid value is passed to a string-to-number conversion method.
<a href="#"><u>RuntimeException</u></a>	<code>RuntimeException</code> is the superclass of those exceptions that can be thrown during the normal operation of the Java Virtual Machine.	<a href="#"><u>RuntimeException</u></a>	<code>RuntimeException</code> is the superclass of all classes that represent exceptional conditions which occur as a result of executing an application in the virtual machine.
<a href="#"><u>SecurityException</u></a>	Thrown by the security manager to indicate a security violation.	<a href="#"><u>SecurityException</u></a>	Thrown when a security manager check fails.
<a href="#"><u>StringIndexOutOfBoundsException</u></a>	Thrown by <code>String</code> methods to indicate that an index is either negative or greater than the size	<a href="#"><u>StringIndexOutOfBoundsException</u></a>	Thrown when the a string is indexed with a value less than zero, or greater than or equal to the size of the

Java™ 2 Platform Standard Edition 5.0 API Specification (java.lang)		Android APIs (java.lang)																									
<a href="#">Exception</a>	of the string.	<a href="#">BoundsException</a>	array.																								
<a href="#">TypeNotPresentException</a>	Thrown when an application tries to access a type using a string representing the type's name, but no definition for the type with the specified name can be found.	<a href="#">TypeNotPresentException</a>	Thrown when a program tries to access a class, nterface, enum or annotation type through a string that contains the type's name and the type cannot be found.																								
<a href="#">UnsupportedOperationException</a>	Thrown to indicate that the requested operation is not supported.	<a href="#">UnsupportedOperationException</a>	Thrown when an unsupported operation is attempted.																								
<div>Error Summary</div> <table><tr><td><a href="#">AbstractMethodError</a></td><td>Thrown when an application tries to call an abstract method.</td></tr><tr><td><a href="#">AssertionError</a></td><td>Thrown to indicate that an assertion has failed.</td></tr><tr><td><a href="#">ClassCircularityError</a></td><td>Thrown when a circularity has been detected while initializing a class.</td></tr><tr><td><a href="#">ClassFormatError</a></td><td>Thrown when the Java Virtual Machine attempts to read a class file and determines that the file is malformed or otherwise cannot be interpreted as a class file.</td></tr><tr><td><a href="#">Error</a></td><td>An <code>Error</code> is a subclass of <code>Throwable</code> that indicates serious problems that a reasonable application should not try to catch.</td></tr><tr><td><a href="#">ExceptionInInitializerError</a></td><td>Signals that an unexpected exception has occurred in a static initializer.</td></tr></table>		<a href="#">AbstractMethodError</a>	Thrown when an application tries to call an abstract method.	<a href="#">AssertionError</a>	Thrown to indicate that an assertion has failed.	<a href="#">ClassCircularityError</a>	Thrown when a circularity has been detected while initializing a class.	<a href="#">ClassFormatError</a>	Thrown when the Java Virtual Machine attempts to read a class file and determines that the file is malformed or otherwise cannot be interpreted as a class file.	<a href="#">Error</a>	An <code>Error</code> is a subclass of <code>Throwable</code> that indicates serious problems that a reasonable application should not try to catch.	<a href="#">ExceptionInInitializerError</a>	Signals that an unexpected exception has occurred in a static initializer.	<div>Errors</div> <table><tr><td><a href="#">AbstractMethodError</a></td><td>Thrown by the virtual machine when an abstract method is called.</td></tr><tr><td><a href="#">AssertionError</a></td><td>Thrown when an assertion has failed.</td></tr><tr><td><a href="#">ClassCircularityError</a></td><td>Thrown when the virtual machine notices that an attempt is made to load a class which would directly or ndirectly inherit from one of its subclasses.</td></tr><tr><td><a href="#">ClassFormatError</a></td><td>Thrown by a class loader when a class file has an llegal format or if the data that it contains can not be nterpreted as a class.</td></tr><tr><td><a href="#">Error</a></td><td><code>Error</code> is the superclass of all classes that represent unrecoverable errors.</td></tr><tr><td><a href="#">ExceptionInInitializerError</a></td><td>Thrown when an exception occurs during class nitialization.</td></tr></table>		<a href="#">AbstractMethodError</a>	Thrown by the virtual machine when an abstract method is called.	<a href="#">AssertionError</a>	Thrown when an assertion has failed.	<a href="#">ClassCircularityError</a>	Thrown when the virtual machine notices that an attempt is made to load a class which would directly or ndirectly inherit from one of its subclasses.	<a href="#">ClassFormatError</a>	Thrown by a class loader when a class file has an llegal format or if the data that it contains can not be nterpreted as a class.	<a href="#">Error</a>	<code>Error</code> is the superclass of all classes that represent unrecoverable errors.	<a href="#">ExceptionInInitializerError</a>	Thrown when an exception occurs during class nitialization.
<a href="#">AbstractMethodError</a>	Thrown when an application tries to call an abstract method.																										
<a href="#">AssertionError</a>	Thrown to indicate that an assertion has failed.																										
<a href="#">ClassCircularityError</a>	Thrown when a circularity has been detected while initializing a class.																										
<a href="#">ClassFormatError</a>	Thrown when the Java Virtual Machine attempts to read a class file and determines that the file is malformed or otherwise cannot be interpreted as a class file.																										
<a href="#">Error</a>	An <code>Error</code> is a subclass of <code>Throwable</code> that indicates serious problems that a reasonable application should not try to catch.																										
<a href="#">ExceptionInInitializerError</a>	Signals that an unexpected exception has occurred in a static initializer.																										
<a href="#">AbstractMethodError</a>	Thrown by the virtual machine when an abstract method is called.																										
<a href="#">AssertionError</a>	Thrown when an assertion has failed.																										
<a href="#">ClassCircularityError</a>	Thrown when the virtual machine notices that an attempt is made to load a class which would directly or ndirectly inherit from one of its subclasses.																										
<a href="#">ClassFormatError</a>	Thrown by a class loader when a class file has an llegal format or if the data that it contains can not be nterpreted as a class.																										
<a href="#">Error</a>	<code>Error</code> is the superclass of all classes that represent unrecoverable errors.																										
<a href="#">ExceptionInInitializerError</a>	Thrown when an exception occurs during class nitialization.																										

Java™ 2 Platform Standard Edition 5.0 API Specification (java.lang)		Android APIs (java.lang)	
<a href="#"><u>IllegalAccessError</u></a>	Thrown if an application attempts to access or modify a field, or to call a method that it does not have access to.	<a href="#"><u>IllegalAccessError</u></a>	Thrown when the virtual machine notices that a program tries access a field which is not accessible from where it is referenced.
<a href="#"><u>IncompatibleClassChangeError</u></a>	Thrown when an incompatible class change has occurred to some class definition.	<a href="#"><u>IncompatibleClassChangeError</u></a>	<a href="#"><u>IncompatibleClassChangeError</u></a> is the superclass of all classes which represent errors that occur when inconsistent class files are loaded into the same running image.
<a href="#"><u>InstantiationError</u></a>	Thrown when an application tries to use the Java <code>new</code> construct to instantiate an abstract class or an interface.	<a href="#"><u>InstantiationError</u></a>	Thrown when the virtual machine notices that a program tries to create a new instance of a class which has no visible constructors from the location where <code>new</code> is invoked.
<a href="#"><u>InternalError</u></a>	Thrown to indicate some unexpected internal error has occurred in the Java Virtual Machine.	<a href="#"><u>InternalError</u></a>	Thrown when the virtual machine notices that it has gotten into an undefined state.
<a href="#"><u>LinkageError</u></a>	Subclasses of <code>LinkageError</code> indicate that a class has some dependency on another class; however, the latter class has incompatibly changed after the compilation of the former class.	<a href="#"><u>LinkageError</u></a>	<a href="#"><u>LinkageError</u></a> is the superclass of all error classes that occur when loading and linking class files.
<a href="#"><u>NoClassDefFoundError</u></a>	Thrown if the Java Virtual Machine or a <code>ClassLoader</code> instance tries to load in the definition of a class (as part of a normal method call or as part of creating a new instance using the <code>new</code> expression) and no definition of the class could be found.	<a href="#"><u>NoClassDefFoundError</u></a>	Thrown when the virtual machine is unable to locate a class which it has been asked to load.
<a href="#"><u>NoSuchFieldError</u></a>	Thrown if an application tries to access or modify a specified field of an object, and that object no longer has that field.	<a href="#"><u>NoSuchFieldError</u></a>	Thrown when the virtual machine notices that a program tries to reference, on a class or object, a field that does not exist.

Java™ 2 Platform Standard Edition 5.0 API Specification (java.lang)		Android APIs (java.lang)	
<a href="#"><u>NoSuchMethodError</u></a>	Thrown if an application tries to call a specified method of a class (either static or instance), and that class no longer has a definition of that method.	<a href="#"><u>NoSuchMethodError</u></a>	Thrown when the virtual machine notices that a program tries to reference, on a class or object, a method that does not exist.
<a href="#"><u>OutOfMemoryError</u></a>	Thrown when the Java Virtual Machine cannot allocate an object because it is out of memory, and no more memory could be made available by the garbage collector.	<a href="#"><u>OutOfMemoryError</u></a>	Thrown when a request for memory is made that can not be satisfied using the available platform resources.
<a href="#"><u>StackOverflowError</u></a>	Thrown when a stack overflow occurs because an application recurses too deeply.	<a href="#"><u>StackOverflowError</u></a>	Thrown when the depth of the callstack of the running program exceeds some platform or virtual machine specific limit.
<a href="#"><u>ThreadDeath</u></a>	An instance of <code>ThreadDeath</code> is thrown in the victim thread when the <code>stop</code> method with zero arguments in class <code>Thread</code> is called.	<a href="#"><u>ThreadDeath</u></a>	<code>ThreadDeath</code> is thrown when a thread stops executing.
<a href="#"><u>UnknownError</u></a>	Thrown when an unknown but serious exception has occurred in the Java Virtual Machine.	<a href="#"><u>UnknownError</u></a>	Thrown when the virtual machine must throw an error which does not match any known exceptional condition.
<a href="#"><u>UnsatisfiedLinkError</u></a>	Thrown if the Java Virtual Machine cannot find an appropriate native-language definition of a method declared <code>native</code> .	<a href="#"><u>UnsatisfiedLinkError</u></a>	Thrown when an attempt is made to invoke a native for which an implementation could not be found.
<a href="#"><u>UnsupportedClassVersionError</u></a>	Thrown when the Java Virtual Machine attempts to read a class file and determines that the major and minor version numbers in the file are not supported.	<a href="#"><u>UnsupportedClassVersionError</u></a>	Thrown when an attempt is made to load a class with a format version that is not supported by the virtual machine.
<a href="#"><u>VerifyError</u></a>	Thrown when the "verifier" detects that a class file, though well formed, contains some sort of internal inconsistency or security problem.	<a href="#"><u>VerifyError</u></a>	Thrown when the virtual machine notices that an attempt is made to load a class which does not pass the class verification phase.
<a href="#"><u>VirtualMachineError</u></a>	Thrown to indicate that the Java Virtual Machine is broken or has run out of resources	<a href="#"><u>VirtualMachineError</u></a>	<a href="#"><u>VirtualMachineError</u></a> is the superclass of all error classes that occur during the operation of the virtual machine.

Java™ 2 Platform Standard Edition 5.0 API Specification (java.lang)		Android APIs (java.lang)
	necessary for it to continue operating.	
<b>Annotation Types Summary</b>		
<a href="#"><u>Deprecated</u></a>	A program element annotated @Deprecated is one that programmers are discouraged from using, typically because it is dangerous, or because a better alternative exists.	
<a href="#"><u>Override</u></a>	Indicates that a method declaration is intended to override a method declaration in a superclass.	
<a href="#"><u>Suppress Warnings</u></a>	Indicates that the named compiler warnings should be suppressed in the annotated element (and in all program elements contained in the annotated element).	



## Exhibit Copyright-C

Java™ 2 Platform Standard Edition 5.0 API Specification (java io)		Android APIs (java.io)	
<b>Interface Summary</b>		<b>Interfaces</b>	
<a href="#"><u>Closeable</u></a>	A <code>Closeable</code> is a source or destination of data that can be closed.	<a href="#"><u>Closeable</u></a>	Defines an interface for classes that can (or need to) be closed once they are not used any longer.
<a href="#"><u>DataInput</u></a>	The <code>DataInput</code> interface provides for reading bytes from a binary stream and reconstructing from them data in any of the Java primitive types.	<a href="#"><u>DataInput</u></a>	Defines an interface for classes that are able to read typed data from some source.
<a href="#"><u>DataOutput</u></a>	The <code>DataOutput</code> interface provides for converting data from any of the Java primitive types to a series of bytes and writing these bytes to a binary stream.	<a href="#"><u>DataOutput</u></a>	Defines an interface for classes that are able to write typed data to some target.
<a href="#"><u>Externalizable</u></a>	Only the identity of the class of an <code>Externalizable</code> instance is written in the serialization stream and it is the responsibility of the class to save and restore the contents of its instances.	<a href="#"><u>Externalizable</u></a>	Defines an interface for classes that want to be serializable, but have their own binary representation.
<a href="#"><u>FileFilter</u></a>	A filter for abstract pathnames.	<a href="#"><u>FileFilter</u></a>	An interface for filtering <code>File</code> objects based on their names or other information.
<a href="#"><u>FilenameFilter</u></a>	Instances of classes that implement this interface are used to filter filenames.	<a href="#"><u>FilenameFilter</u></a>	An interface for filtering <code>File</code> objects based on their names or the directory they reside in.



Java™ 2 Platform Standard Edition 5.0 API Specification (java.io)		Android APIs (java.io)	
<a href="#"><u>Flushable</u></a>	A <code>Flushable</code> is a destination of data that can be flushed.	<a href="#"><u>Flushable</u></a>	Defines an interface for classes that can (or need to) be flushed, typically before some output processing is considered to be finished and the object gets closed.
<a href="#"><u>ObjectInput</u></a>	<code>ObjectInput</code> extends the <code>DataInput</code> interface to include the reading of objects.	<a href="#"><u>ObjectInput</u></a>	Defines an interface for classes that allow reading serialized objects.
<a href="#"><u>ObjectInputValidation</u></a>	Callback interface to allow validation of objects within a graph.	<a href="#"><u>ObjectInputValidation</u></a>	A callback interface for post-deserialization checks on objects.
<a href="#"><u>ObjectOutput</u></a>	<code>ObjectOutput</code> extends the <code>DataOutput</code> interface to include writing of objects.	<a href="#"><u>ObjectOutput</u></a>	Defines an interface for classes that allow reading serialized objects.
<a href="#"><u>ObjectStreamConstants</u></a>	Constants written into the Object Serialization Stream.	<a href="#"><u>ObjectStreamConstants</u></a>	A helper interface with constants used by the serialization implementation.
<a href="#"><u>Serializable</u></a>	Serializability of a class is enabled by the class implementing the <code>java.io.Serializable</code> interface.	<a href="#"><u>Serializable</u></a>	An empty marker interface for classes that want to support serialization and deserialization based on the <code>ObjectOutputStream</code> and <code>ObjectInputStream</code> classes.
<b>Class Summary</b>		<b>Classes</b>	
<a href="#"><u>BufferedInputStream</u></a>	A <code>BufferedInputStream</code> adds functionality to another input stream—namely, the ability to buffer the input and to support the <code>mark</code> and <code>reset</code> methods.	<a href="#"><u>BufferedInputStream</u></a>	Wraps an existing <code>InputStream</code> and <i>buffers</i> the input.
<a href="#"><u>BufferedOutputStream</u></a>	The class implements a buffered output stream.	<a href="#"><u>BufferedOutputStream</u></a>	Wraps an existing <code>OutputStream</code> and <i>buffers</i> the output.
<a href="#"><u>BufferedReader</u></a>	Read text from a character-input stream, buffering characters so as to provide for the efficient reading of characters, arrays,	<a href="#"><u>BufferedReader</u></a>	Wraps an existing <code>Reader</code> and <i>buffers</i> the input.

Java™ 2 Platform Standard Edition 5.0 API Specification (java.io)		Android APIs (java.io)	
	and lines.		
<a href="#"><u>BufferedWriter</u></a>	Write text to a character-output stream, buffering characters so as to provide for the efficient writing of single characters, arrays, and strings.	<a href="#">BufferedWriter</a>	Wraps an existing <a href="#">Writer</a> and <i>buffers</i> the output.
<a href="#"><u>ByteArrayInputStream</u></a>	A <code>ByteArrayInputStream</code> contains an internal buffer that contains bytes that may be read from the stream.	<a href="#">ByteArrayInputStream</a>	A specialized <a href="#">InputStream</a> for reading the contents of a byte array.
<a href="#"><u>ByteArrayOutputStream</u></a>	This class implements an output stream in which the data is written into a byte array.	<a href="#">ByteArrayOutputStream</a>	A specialized <a href="#">OutputStream</a> for class for writing content to an (internal) byte array.
<a href="#"><u>CharArrayReader</u></a>	This class implements a character buffer that can be used as a character-input stream.	<a href="#">CharArrayReader</a>	A specialized <a href="#">Reader</a> for reading the contents of a char array.
<a href="#"><u>CharArrayWriter</u></a>	This class implements a character buffer that can be used as an <code>Writer</code> .	<a href="#">CharArrayWriter</a>	A specialized <a href="#">Writer</a> for class for writing content to an (internal) char array.
<a href="#"><u>DataInputStream</u></a>	A data input stream lets an application read primitive Java data types from an underlying input stream in a machine-independent way.	<a href="#">Console</a>	Provides access to the console, if available.
<a href="#"><u>DataOutputStream</u></a>	A data output stream lets an application write primitive Java data types to an output stream in a portable way.	<a href="#">DataInputStream</a>	Wraps an existing <a href="#">InputStream</a> and reads typed data from it.
<a href="#"><u>File</u></a>	An abstract representation of file and directory pathnames.	<a href="#">DataOutputStream</a>	Wraps an existing <a href="#">OutputStream</a> and writes typed data to it.
<a href="#"><u>FileDescriptor</u></a>	Instances of the file descriptor class serve	<a href="#">File</a>	An "abstract" representation of a file system entity identified by a pathname.

Java™ 2 Platform Standard Edition 5.0 API Specification (java.io)		Android APIs (java.io)	
	as an opaque handle to the underlying machine-specific structure representing an open file, an open socket, or another source or sink of bytes.	<a href="#">FileDescriptor</a>	The lowest-level representation of a file, device, or socket.
<a href="#">FileInputStream</a>	A <code>FileInputStream</code> obtains input bytes from a file in a file system.	<a href="#">FileInputStream</a>	A specialized <a href="#">InputStream</a> that reads from a file in the file system.
<a href="#">FileOutputStream</a>	A file output stream is an output stream for writing data to a <code>File</code> or to a <code>FileDescriptor</code> .	<a href="#">FileOutputStream</a>	A specialized <a href="#">OutputStream</a> that writes to a file in the file system.
<a href="#">FilePermission</a>	This class represents access to a file or directory.	<a href="#">FilePermission</a>	A permission for accessing a file or directory.
<a href="#">FileReader</a>	Convenience class for reading character files.	<a href="#">FileReader</a>	A specialized <a href="#">Reader</a> that reads from a file in the file system.
<a href="#">FileWriter</a>	Convenience class for writing character files.	<a href="#">FileWriter</a>	A specialized <a href="#">Writer</a> that writes to a file in the file system.
<a href="#">FilterInputStream</a>	A <code>FilterInputStream</code> contains some other input stream, which it uses as its basic source of data, possibly transforming the data along the way or providing additional functionality.	<a href="#">FilterInputStream</a>	Wraps an existing <a href="#">InputStream</a> and performs some transformation on the input data while it is being read.
<a href="#">FilterOutputStream</a>	This class is the superclass of all classes that filter output streams.	<a href="#">FilterOutputStream</a>	Wraps an existing <a href="#">OutputStream</a> and performs some transformation on the output data while it is being written.

Java™ 2 Platform Standard Edition 5.0 API Specification (java.io)		Android APIs (java.io)	
<a href="#"><u>FilterReader</u></a>	Abstract class for reading filtered character streams.	<a href="#"><u>FilterReader</u></a>	Wraps an existing <a href="#"><u>Reader</u></a> and performs some transformation on the input data while it is being read.
<a href="#"><u>FilterWriter</u></a>	Abstract class for writing filtered character streams.	<a href="#"><u>FilterWriter</u></a>	Wraps an existing <a href="#"><u>Writer</u></a> and performs some transformation on the output data while it is being written.
<a href="#"><u>InputStream</u></a>	This abstract class is the superclass of all classes representing an input stream of bytes.	<a href="#"><u>nputStream</u></a>	The base class for all input streams.
<a href="#"><u>InputStreamReader</u></a>	An <a href="#"><u>InputStreamReader</u></a> is a bridge from byte streams to character streams: It reads bytes and decodes them into characters using a specified <a href="#"><u>charset</u></a> .	<a href="#"><u>nputStreamReader</u></a>	A class for turning a byte stream into a character stream.
<a href="#"><u>LineNumberInputStream</u></a>	<b>Deprecated.</b> <i>This class incorrectly assumes that bytes adequately represent characters.</i>	<a href="#"><u>LineNumberInputStream</u></a>	<i>This class is deprecated. Use <a href="#"><u>LineNumberReader</u></a></i>
<a href="#"><u>LineNumberReader</u></a>	A buffered character-input stream that keeps track of line numbers.	<a href="#"><u>LineNumberReader</u></a>	Wraps an existing <a href="#"><u>Reader</u></a> and counts the line terminators encountered while reading the data.
<a href="#"><u>ObjectInputStream</u></a>	An <a href="#"><u>ObjectInputStream</u></a> deserializes primitive data and objects previously written using an <a href="#"><u>ObjectOutputStream</u></a> .	<a href="#"><u>ObjectInputStream</u></a>	A specialized <a href="#"><u>InputStream</u></a> that is able to read (deserialize) Java objects as well as primitive data types (int, byte, char etc.).
<a href="#"><u>ObjectInputStream.GetField</u></a>	Provide access to the persistent fields read from the input stream.	<a href="#"><u>ObjectInputStream.GetField</u></a>	<a href="#"><u>GetField</u></a> is an inner class that provides access to the persistent fields read from the source stream.

Java™ 2 Platform Standard Edition 5.0 API Specification (java.io)		Android APIs (java.io)	
<a href="#"><u>ObjectOutputStream</u></a>	An ObjectOutputStream writes primitive data types and graphs of Java objects to an OutputStream.	<a href="#"><u>ObjectOutputStream</u></a>	A specialized <a href="#"><u>OutputStream</u></a> that is able to write (serialize) Java objects as well as primitive data types (int, byte, char etc.).
<a href="#"><u>ObjectOutputStream.PutField</u></a>	Provide programmatic access to the persistent fields to be written to ObjectOutputStream.	<a href="#"><u>ObjectOutputStream.PutField</u></a>	PutField is an inner class to provide access to the persistent fields that are written to the target stream.
<a href="#"><u>ObjectStreamClass</u></a>	Serialization's descriptor for classes.	<a href="#"><u>ObjectStreamClass</u></a>	Represents a descriptor for identifying a class during serialization and deserialization.
<a href="#"><u>ObjectStreamField</u></a>	A description of a Serializable field from a Serializable class.	<a href="#"><u>ObjectStreamField</u></a>	Describes a field for the purpose of serialization.
<a href="#"><u>OutputStream</u></a>	This abstract class is the superclass of all classes representing an output stream of bytes.	<a href="#"><u>OutputStream</u></a>	The base class for all output streams.
<a href="#"><u>OutputStreamWriter</u></a>	An OutputStreamWriter is a bridge from character streams to byte streams: Characters written to it are encoded into bytes using a specified <a href="#"><u>charset</u></a> .	<a href="#"><u>OutputStreamWriter</u></a>	A class for turning a character stream into a byte stream.
<a href="#"><u>PipedInputStream</u></a>	A piped input stream should be connected to a piped output stream; the piped input stream then provides whatever data bytes are written to the piped output stream.	<a href="#"><u>PipedInputStream</u></a>	Receives information from a communications pipe.
<a href="#"><u>PipedOutputStream</u></a>	A piped output stream can be connected to a piped input stream to create a communications pipe.	<a href="#"><u>PipedOutputStream</u></a>	Places information on a communications pipe.

Java™ 2 Platform Standard Edition 5.0 API Specification (java.io)		Android APIs (java.io)	
<a href="#"><u>PipedReader</u></a>	Piped character-input streams.	<a href="#"><u>PipedReader</u></a>	Receives information on a communications pipe.
<a href="#"><u>PipedWriter</u></a>	Piped character-output streams.	<a href="#"><u>PipedWriter</u></a>	Places information on a communications pipe.
<a href="#"><u>PrintStream</u></a>	A <code>PrintStream</code> adds functionality to another output stream, namely the ability to print representations of various data values conveniently.	<a href="#"><u>PrintStream</u></a>	Wraps an existing <a href="#"><u>OutputStream</u></a> and provides convenience methods for writing common data types in a human readable format.
<a href="#"><u>PrintWriter</u></a>	Print formatted representations of objects to a text-output stream.	<a href="#"><u>PrintWriter</u></a>	Wraps either an existing <a href="#"><u>OutputStream</u></a> or an existing <a href="#"><u>Writer</u></a> and provides convenience methods for printing common data types in a human readable format.
<a href="#"><u>PushbackInputStream</u></a>	A <code>PushbackInputStream</code> adds functionality to another input stream, namely the ability to "push back" or "unread" one byte.	<a href="#"><u>PushbackInputStream</u></a>	Wraps an existing <a href="#"><u>InputStream</u></a> and adds functionality to "push back" bytes that have been read, so that they can be read again.
<a href="#"><u>PushbackReader</u></a>	A character-stream reader that allows characters to be pushed back into the stream.	<a href="#"><u>PushbackReader</u></a>	Wraps an existing <a href="#"><u>Reader</u></a> and adds functionality to "push back" characters that have been read, so that they can be

Java™ 2 Platform Standard Edition 5.0 API Specification (java.io)		Android APIs (java.io)	
<a href="#"><u>RandomAccessFile</u></a>	Instances of this class support both reading and writing to a random access file.		read again.
<a href="#"><u>Reader</u></a>	Abstract class for reading character streams.	<a href="#"><u>RandomAccessFile</u></a>	Allows reading from and writing to a file in a random-access manner.
<a href="#"><u>SequenceInputStream</u></a>	A <code>SequenceInputStream</code> represents the logical concatenation of other input streams.	<a href="#"><u>Reader</u></a>	The base class for all readers.
<a href="#"><u>SerializablePermission</u></a>	This class is for Serializable permissions.	<a href="#"><u>SequenceInputStream</u></a>	Concatenates two or more existing <code>InputStreamS</code> .
<a href="#"><u>StreamTokenizer</u></a>	The <code>StreamTokenizer</code> class takes an input stream and parses it into "tokens", allowing the tokens to be read one at a time.	<a href="#"><u>SerializablePermission</u></a>	Is used to enable access to potentially unsafe serialization operations.
<a href="#"><u>StringBufferInputStream</u></a>	<b>Deprecated.</b> <i>This class does not properly convert characters into bytes.</i>	<a href="#"><u>StreamTokenizer</u></a>	Parses a stream into a set of defined tokens, one at a time.
<a href="#"><u>StringReader</u></a>	A character stream whose source is a string.	<a href="#"><u>StringBufferInputStream</u></a>	<i>This class is deprecated. Use <code>StringReader</code></i>
<a href="#"><u>StringWriter</u></a>	A character stream that collects its output in a string buffer, which can then be used to construct a string.	<a href="#"><u>StringReader</u></a>	A specialized <code>Reader</code> that reads characters from a <code>String</code> in a sequential manner.
<a href="#"><u>Writer</u></a>	Abstract class for writing to character streams.	<a href="#"><u>StringWriter</u></a>	A specialized <code>Writer</code> that writes characters to a <code>StringBuffer</code> in a

Java™ 2 Platform Standard Edition 5.0 API Specification (java.io)		Android APIs (java.io)	
			sequential manner, appending them in the process.
		<a href="#">Writer</a>	The base class for all writers.
<b>Exception Summary</b>		<b>Exceptions</b>	
<a href="#">CharConversionException</a>	Base class for character conversion exceptions.	<a href="#">CharConversionException</a>	The top level class for character conversion exceptions.
<a href="#">EOFException</a>	Signals that an end of file or end of stream has been reached unexpectedly during input.	<a href="#">EOFException</a>	Thrown when a program encounters the end of a file or stream during an input operation.
<a href="#">FileNotFoundException</a>	Signals that an attempt to open the file denoted by a specified pathname has failed.	<a href="#">FileNotFoundException</a>	Thrown when a file specified by a program cannot be found.
<a href="#">InterruptedIOException</a>	Signals that an I/O operation has been interrupted.	<a href="#">InterruptedIOException</a>	Signals that a blocking I/O operation has been interrupted.
<a href="#">InvalidClassException</a>	Thrown when the Serialization runtime detects one of the following problems with a Class.	<a href="#">InvalidClassException</a>	Signals a problem during the serialization or or deserialization of an object.
<a href="#">InvalidObjectException</a>	Indicates that one or more deserialized objects failed validation tests.	<a href="#">InvalidObjectException</a>	Signals that, during deserialization, the validation of an object has failed.
<a href="#">IOException</a>	Signals that an I/O exception of some sort has occurred.		



Java™ 2 Platform Standard Edition 5.0 API Specification (java.io)		Android APIs (java.io)	
<a href="#"><u>NotActiveException</u></a>	Thrown when serialization or deserialization is not active.	<a href="#"><u>OException</u></a>	Signals a general, I/O-related error.
<a href="#"><u>NotSerializableException</u></a>	Thrown when an instance is required to have a Serializable interface.	<a href="#"><u>NotActiveException</u></a>	Signals that a serialization-related method has been invoked in the wrong place.
<a href="#"><u>ObjectStreamException</u></a>	Superclass of all exceptions specific to Object Stream classes.	<a href="#"><u>NotSerializableException</u></a>	Signals that an object that is not serializable has been passed into the <code>ObjectOutput.writeObject()</code> method.
<a href="#"><u>OptionalDataException</u></a>	Exception indicating the failure of an object read operation due to unread primitive data, or the end of data belonging to a serialized object in the stream.	<a href="#"><u>ObjectStreamException</u></a>	Signals some sort of problem during either serialization or deserialization of objects.
<a href="#"><u>StreamCorruptedException</u></a>	Thrown when control information that was read from an object stream violates internal consistency checks.	<a href="#"><u>OptionalDataException</u></a>	Signals that the <code>ObjectInputStream</code> class encountered a primitive type ( <code>int</code> , <code>char</code> etc.) instead of an object instance in the input stream.
<a href="#"><u>SyncFailedException</u></a>	Signals that a sync operation has failed.	<a href="#"><u>StreamCorruptedException</u></a>	Signals that the <code>readObject()</code> method could not read an object due to missing information (for example, a cyclic reference that doesn't match a previous instance, or a missing class descriptor for the object to be loaded).
<a href="#"><u>UnsupportedEncodingException</u></a>	The Character Encoding is not supported.	<a href="#"><u>SyncFailedException</u></a>	Signals that the <code>sync()</code> method has failed to complete.
<a href="#"><u>UTFDataFormatException</u></a>	Signals that a malformed string in <a href="#"><u>modified UTF-8</u></a> format has been read in a data input stream or by any class	<a href="#"><u>UnsupportedEncodingException</u></a>	Thrown when a program asks for a particular character converter that is unavailable.

Java™ 2 Platform Standard Edition 5.0 API Specification (java.io)		Android APIs (java.io)	
	that implements the data input interface.	<a href="#">UTFDataFormatException</a>	Signals that an incorrectly encoded UTF-8 string has been encountered, most likely while reading some <a href="#">DataInputStream</a> .
<a href="#">WriteAbortedException</a>	Signals that one of the <code>ObjectStreamExceptions</code> was thrown during a write operation.	<a href="#">WriteAbortedException</a>	Signals that the <code>readObject()</code> method has detected an exception marker in the input stream.

## Exhibit Copyright-D

Java™ 2 Platform Standard Edition 5.0 API Specification (java.security)		Android APIs (java.security)	
<b>Interface Summary</b>		<b>Interfaces</b>	
<a href="#"><u>Certificate</u></a>	<b>Deprecated.</b> A new certificate handling package is created in the Java 2 platform.	<a href="#"><u>Certificate</u></a>	<i>This interface is deprecated. Replaced by behavior in <a href="#"><u>java.security.cert</u></a></i>
<a href="#"><u>DomainCombiner</u></a>	A DomainCombiner provides a means to dynamically update the ProtectionDomains associated with the current AccessControlContext.	<a href="#"><u>DomainCombiner</u></a>	<a href="#"><u>DomainCombiner</u></a> is used to update and optimize <a href="#"><u>ProtectionDomains</u></a> from an <a href="#"><u>AccessControlContext</u></a> .
<a href="#"><u>Guard</u></a>	This interface represents a guard, which is an object that is used to protect access to another object.	<a href="#"><u>Guard</u></a>	<a href="#"><u>Guard</u></a> implementors protect access to other objects.
<a href="#"><u>Key</u></a>	The Key interface is the top-level interface for all keys.	<a href="#"><u>Key</u></a>	<a href="#"><u>Key</u></a> is the common interface for all keys.
<a href="#"><u>KeyStore.Entry</u></a>	A marker interface for KeyStore entry types.	<a href="#"><u>KeyStore.Entry</u></a>	<a href="#"><u>Entry</u></a> is the common marker interface for a <a href="#"><u>KeyStore</u></a> entry.
<a href="#"><u>KeyStore.LoadStoreParameter</u></a>	A marker interface for KeyStore <a href="#"><u>load</u></a> and <a href="#"><u>store</u></a> parameters.	<a href="#"><u>KeyStore.LoadStoreParameter</u></a>	<a href="#"><u>LoadStoreParameter</u></a> represents a parameter that specifies how a <a href="#"><u>KeyStore</u></a> can be loaded and stored.
<a href="#"><u>KeyStore.ProtectionParameter</u></a>	A marker interface for keystore protection parameters.	<a href="#"><u>KeyStore.ProtectionParameter</u></a>	<a href="#"><u>ProtectionParameter</u></a> is a marker interface for protection parameters.
<a href="#"><u>Principal</u></a>	This interface represents the abstract notion of a principal, which can be used to	<a href="#"><u>Policy.Parameters</u></a>	A marker interface for Policy parameters.

Java™ 2 Platform Standard Edition 5.0 API Specification (java.security)		Android APIs (java.security)	
	represent any entity, such as an individual, a corporation, and a login id.	Principal	Principals are objects which have identities.
<a href="#">PrivateKey</a>	A private key.	PrivateKey	PrivateKey is the common interface for private keys.
<a href="#">PrivilegedAction&lt;T&gt;</a>	A computation to be performed with privileges enabled.	PrivilegedAction<T>	PrivilegedAction represents an action that can be executed privileged regarding access control.
<a href="#">PrivilegedExceptionAction&lt;T&gt;</a>	A computation to be performed with privileges enabled, that throws one or more checked exceptions.	PrivilegedExceptionAction<T>	PrivilegedAction represents an action, that can be executed privileged regarding access control.
<a href="#">PublicKey</a>	A public key.	PublicKey	PublicKey is the common interface for public keys.
<b>Class Summary</b>		<b>Classes</b>	
<a href="#">AccessControlContext</a>	An AccessControlContext is used to make system resource access decisions based on the context it encapsulates.	AccessControlContext	AccessControlContext encapsulates the ProtectionDomains on which access control decisions are based.
		AccessController	AccessController provides static methods to perform access control checks and privileged operations.

Java™ 2 Platform Standard Edition 5.0 API Specification (java.security)		Android APIs (java.security)	
<a href="#"><u>AlgorithmParameterGenerator</u></a>	The AlgorithmParameterGenerator class is used to generate a set of parameters to be used with a certain algorithm.	AlgorithmParameterGenerator	AlgorithmParameterGenerator is an engine class which is capable of generating parameters for the algorithm it was initialized with.
<a href="#"><u>AlgorithmParameterGeneratorSpi</u></a>	This class defines the <i>Service Provider Interface (SPI)</i> for the AlgorithmParameterGenerator class, which is used to generate a set of parameters to be used with a certain algorithm.	AlgorithmParameterGeneratorSpi	AlgorithmParameterGeneratorSpi is the Service Provider Interface (SPI) definition for AlgorithmParameterGenerator.
<a href="#"><u>AlgorithmParameters</u></a>	This class is used as an opaque representation of cryptographic parameters.	AlgorithmParameters	AlgorithmParameters is an engine class which provides algorithm parameters.
<a href="#"><u>AlgorithmParametersSpi</u></a>	This class defines the <i>Service Provider Interface (SPI)</i> for the AlgorithmParameters class, which is used to manage algorithm parameters.	AlgorithmParametersSpi	AlgorithmParametersSpi is the Service Provider Interface (SPI) definition for AlgorithmParameters.
<a href="#"><u>AllPermission</u></a>	The AllPermission is a permission that implies all other permissions.	AllPermission	AllPermission represents the permission to perform any operation.
<a href="#"><u>AuthProvider</u></a>	This class defines login and logout methods for a provider.	AuthProvider	AuthProvider is an abstract superclass for Java Security Provider which provide login and logout.
<a href="#"><u>BasicPermission</u></a>	The BasicPermission class extends the Permission class, and can be used as the base class for permissions that want to follow the same naming convention as BasicPermission.	BasicPermission	BasicPermission is the common base class of all permissions which have a name but no action lists.
<a href="#"><u>CodeSigner</u></a>	This class encapsulates information about a code signer.	CodeSigner	CodeSigner represents a signer of code.

Java™ 2 Platform Standard Edition 5.0 API Specification (java.security)		Android APIs (java.security)	
<a href="#"><u>CodeSource</u></a>	This class extends the concept of a codebase to encapsulate not only the location (URL) but also the certificate chains that were used to verify signed code originating from that location.	<a href="#"><u>CodeSource</u></a>	<a href="#"><u>CodeSource</u></a> encapsulates the location from where code is loaded and the certificates that were used to verify that code.
<a href="#"><u>DigestInputStream</u></a>	A transparent stream that updates the associated message digest using the bits going through the stream.	<a href="#"><u>DigestInputStream</u></a>	<a href="#"><u>DigestInputStream</u></a> is a <a href="#"><u>FilterInputStream</u></a> which maintains an associated message digest.
<a href="#"><u>DigestOutputStream</u></a>	A transparent stream that updates the associated message digest using the bits going through the stream.	<a href="#"><u>DigestOutputStream</u></a>	<a href="#"><u>DigestOutputStream</u></a> is a <a href="#"><u>FilterOutputStream</u></a> which maintains an associated message digest.
<a href="#"><u>GuardedObject</u></a>	A GuardedObject is an object that is used to protect access to another object.	<a href="#"><u>GuardedObject</u></a>	<a href="#"><u>GuardedObject</u></a> controls access to an object, by checking all requests for the object with a <a href="#"><u>Guard</u></a> .
<a href="#"><u>Identity</u></a>	<b>Deprecated.</b> <i>This class is no longer used.</i>	<a href="#"><u>Identity</u></a>	<i>This class is deprecated. The functionality of this class has been replace by <a href="#"><u>Principal</u></a>, <a href="#"><u>KeyStore</u></a> and the <a href="#"><u>java.security.cert</u></a> package.</i>

Java™ 2 Platform Standard Edition 5.0 API Specification (java.security)		Android APIs (java.security)	
<a href="#"><u>IdentityScope</u></a>	<b>Deprecated.</b> <i>This class is no longer used.</i>	<a href="#"><u>IdentityScope</u></a>	<i>This class is deprecated. The functionality of this class has been replaced by <a href="#"><u>Principal</u></a>, <a href="#"><u>KeyStore</u></a> and the <a href="#"><u>java.security.cert</u></a> package.</i>
<a href="#"><u>KeyFactory</u></a>	Key factories are used to convert <i>keys</i> (opaque cryptographic keys of type <code>Key</code> ) into <i>key specifications</i> (transparent representations of the underlying key material), and vice versa.	<a href="#"><u>KeyFactory</u></a>	<a href="#"><u>KeyFactory</u></a> is an engine class that can be used to translate between public and private key objects and convert keys between their external representation, that can be easily transported and their internal representation.
<a href="#"><u>KeyFactorySpi</u></a>	This class defines the <i>Service Provider Interface (SPI)</i> for the <code>KeyFactory</code> class.	<a href="#"><u>KeyFactorySpi</u></a>	<a href="#"><u>KeyFactorySpi</u></a> is the Service Provider Interface (SPI) definition for <a href="#"><u>KeyFactory</u></a> .
<a href="#"><u>KeyPair</u></a>	This class is a simple holder for a key pair (a public key and a private key).	<a href="#"><u>KeyPair</u></a>	<a href="#"><u>KeyPair</u></a> is a container for a public key and a private key.
<a href="#"><u>KeyPairGenerator</u></a>	The <code>KeyPairGenerator</code> class is used to generate pairs of public and private keys.	<a href="#"><u>KeyPairGenerator</u></a>	<a href="#"><u>KeyPairGenerator</u></a> is an engine class which is capable of generating a private key and its related public key utilizing the algorithm it was initialized with.
<a href="#"><u>KeyPairGeneratorSpi</u></a>	This class defines the <i>Service Provider Interface (SPI)</i> for the <code>KeyPairGenerator</code> class, which is used to generate pairs of public and private keys.	<a href="#"><u>KeyPairGeneratorSpi</u></a>	<a href="#"><u>KeyPairGeneratorSpi</u></a> is the Service Provider Interface (SPI) definition for <a href="#"><u>KeyPairGenerator</u></a> .
<a href="#"><u>KeyRep</u></a>	Standardized representation for serialized <code>Key</code> objects.	<a href="#"><u>KeyRep</u></a>	<a href="#"><u>KeyRep</u></a> is a standardized representation

Java™ 2 Platform Standard Edition 5.0 API Specification (java.security)		Android APIs (java.security)	
<a href="#"><u>KeyStore</u></a>	This class represents a storage facility for cryptographic keys and certificates.		for serialized <a href="#"><u>key</u></a> objects.
<a href="#"><u>KeyStore.Builder</u></a>	A description of a to-be-instantiated KeyStore object.	<a href="#"><u>KeyStore</u></a>	<a href="#"><u>KeyStore</u></a> is responsible for maintaining cryptographic keys and their owners.
<a href="#"><u>KeyStore.CallbackHandlerProtection</u></a>	A ProtectionParameter encapsulating a CallbackHandler.	<a href="#"><u>KeyStore.Builder</u></a>	<a href="#"><u>Builder</u></a> is used to construct new instances of <a href="#"><u>KeyStore</u></a> .
<a href="#"><u>KeyStore.PasswordProtection</u></a>	A password-based implementation of ProtectionParameter.	<a href="#"><u>KeyStore.CallbackHandlerProtection</u></a>	<a href="#"><u>CallbackHandlerProtection</u></a> is a <a href="#"><u>ProtectionParameter</u></a> that encapsulates a <a href="#"><u>CallbackHandler</u></a> .
<a href="#"><u>KeyStore.PrivateKeyEntry</u></a>	A KeyStore entry that holds a PrivateKey and corresponding certificate chain.	<a href="#"><u>KeyStore.PasswordProtection</u></a>	<a href="#"><u>PasswordProtection</u></a> is a <a href="#"><u>ProtectionParameter</u></a> that protects a <a href="#"><u>KeyStore</u></a> using a password.
<a href="#"><u>KeyStore.SecretKeyEntry</u></a>	A KeyStore entry that holds a SecretKey.	<a href="#"><u>KeyStore.PrivateKeyEntry</u></a>	<a href="#"><u>PrivateKeyEntry</u></a> represents a <a href="#"><u>KeyStore</u></a> entry that holds a private key.
<a href="#"><u>KeyStore.TrustedCertificateEntry</u></a>	A KeyStore entry that holds a trusted Certificate.	<a href="#"><u>KeyStore.SecretKeyEntry</u></a>	<a href="#"><u>SecretKeyEntry</u></a> represents a <a href="#"><u>KeyStore</u></a> entry that holds a secret key.
		<a href="#"><u>KeyStore.TrustedCertificateEntry</u></a>	<a href="#"><u>TrustedCertificateEntry</u></a> represents a <a href="#"><u>KeyStore</u></a> entry that holds a trusted certificate.



Java™ 2 Platform Standard Edition 5.0 API Specification (java.security)		Android APIs (java.security)	
<a href="#"><u>KeyStoreSpi</u></a>	This class defines the <i>Service Provider Interface (SPI)</i> for the <code>KeyStore</code> class.	<a href="#"><u>KeyStoreSpi</u></a>	<code>KeyStoreSpi</code> is the Service Provider Interface (SPI) definition for <code>KeyStore</code> .
<a href="#"><u>MessageDigest</u></a>	This <code>MessageDigest</code> class provides applications the functionality of a message digest algorithm, such as MD5 or SHA.	<a href="#"><u>MessageDigest</u></a>	Uses a one-way hash function to turn an arbitrary number of bytes into a fixed-length byte sequence.
<a href="#"><u>MessageDigestSpi</u></a>	This class defines the <i>Service Provider Interface (SPI)</i> for the <code>MessageDigest</code> class, which provides the functionality of a message digest algorithm, such as MD5 or SHA.	<a href="#"><u>MessageDigestSpi</u></a>	<code>MessageDigestSpi</code> is the Service Provider Interface (SPI) definition for <code>MessageDigest</code> .
<a href="#"><u>Permission</u></a>	Abstract class for representing access to a system resource.	<a href="#"><u>Permission</u></a>	<code>Permission</code> is the common base class of all permissions that participate in the access control security framework around <code>AccessController</code> and <code>AccessControlContext</code> .
<a href="#"><u>PermissionCollection</u></a>	Abstract class representing a collection of <code>Permission</code> objects.	<a href="#"><u>PermissionCollection</u></a>	<code>PermissionCollection</code> is the common base class for all collections that provide a convenient method for determining whether or not a given permission is implied by any of the permissions present in this collection.
<a href="#"><u>Permissions</u></a>	This class represents a heterogeneous collection of <code>Permissions</code> .	<a href="#"><u>Permissions</u></a>	<code>Permissions</code> represents a <code>PermissionCollection</code> where the contained permissions can be of different types.
		<a href="#"><u>Policy</u></a>	<code>Policy</code> is the common super type of classes which represent a system security policy.

Java™ 2 Platform Standard Edition 5.0 API Specification (java.security)		Android APIs (java.security)	
<a href="#">Policy</a>	This is an abstract class for representing the system security policy for a Java application environment (specifying which permissions are available for code from various sources).	<a href="#">PolicySpi</a>	Represents the Service Provider Interface (SPI) for java.security.Policy class.
<a href="#">ProtectionDomain</a>	This ProtectionDomain class encapsulates the characteristics of a domain, which encloses a set of classes whose instances are granted a set of permissions when being executed on behalf of a given set of Principals.	<a href="#">ProtectionDomain</a>	<a href="#">ProtectionDomain</a> represents all permissions that are granted to a specific code source.
<a href="#">Provider</a>	This class represents a "provider" for the Java Security API, where a provider implements some or all parts of Java Security.	<a href="#">Provider</a>	<a href="#">Provider</a> is the abstract superclass for all security providers in the Java security infrastructure.
<a href="#">Provider.Service</a>	The description of a security service.	<a href="#">Provider.Service</a>	<a href="#">Service</a> represents a service in the Java Security infrastructure.
<a href="#">SecureClassLoader</a>	This class extends ClassLoader with additional support for defining classes with an associated code source and permissions which are retrieved by the system policy by default.	<a href="#">SecureClassLoader</a>	<a href="#">SecureClassLoader</a> represents a <a href="#">ClassLoader</a> which associates the classes it loads with a code source and provide mechanisms to allow the relevant permissions to be retrieved.
<a href="#">SecureRandom</a>	This class provides a cryptographically strong random number generator (RNG).	<a href="#">SecureRandom</a>	This class generates cryptographically secure pseudo-random numbers.
<a href="#">SecureRandomSpi</a>	This class defines the <i>Service Provider Interface (SPI)</i> for the <code>SecureRandom</code> class.	<a href="#">SecureRandomSpi</a>	<a href="#">SecureRandomSpi</a> is the <i>Service Provider Interface (SPI)</i> definition for <a href="#">SecureRandom</a> .
		<a href="#">Security</a>	<a href="#">Security</a> is the central class in the Java Security API.

Java™ 2 Platform Standard Edition 5.0 API Specification (java.security)		Android APIs (java.security)	
<a href="#">Security</a>	This class centralizes all security properties and common security methods.	<a href="#">SecurityPermission</a>	<a href="#">SecurityPermission</a> objects guard access to the mechanisms which implement security.
<a href="#">SecurityPermission</a>	This class is for security permissions.	<a href="#">Signature</a>	<a href="#">Signature</a> is an engine class which is capable of creating and verifying digital signatures, using different algorithms that have been registered with the <a href="#">Security</a> class.
<a href="#">Signature</a>	This Signature class is used to provide applications the functionality of a digital signature algorithm.	<a href="#">SignatureSpi</a>	<a href="#">SignatureSpi</a> is the <i>Service Provider Interface (SPI)</i> definition for <a href="#">Signature</a> .
<a href="#">SignatureSpi</a>	This class defines the <i>Service Provider Interface (SPI)</i> for the <a href="#">Signature</a> class, which is used to provide the functionality of a digital signature algorithm.	<a href="#">SignedObject</a>	A <a href="#">SignedObject</a> instance acts as a container for another object.
<a href="#">SignedObject</a>	<a href="#">SignedObject</a> is a class for the purpose of creating authentic runtime objects whose integrity cannot be compromised without being detected.	<a href="#">Signer</a>	<i>This class is deprecated. Replaced by behavior in <a href="#">java.security.cert</a> package and <a href="#">Principal</a></i>
<a href="#">Signer</a>	<b>Deprecated.</b> <i>This class is no longer used.</i>	<a href="#">Timestamp</a>	<a href="#">Timestamp</a> represents a signed time stamp.
<a href="#">Timestamp</a>	This class encapsulates information about a signed timestamp.	<a href="#">UnresolvedPermission</a>	An <a href="#">UnresolvedPermission</a> represents a <a href="#">Permission</a> whose type should be resolved lazy and not during initialization time of the <a href="#">Policy</a> .
<a href="#">UnresolvedPermission</a>	The <a href="#">UnresolvedPermission</a> class is used to hold Permissions that were "unresolved" when the <a href="#">Policy</a> was initialized.		

Java™ 2 Platform Standard Edition 5.0 API Specification (java.security)		Android APIs (java.security)	
<b>Enum Summary</b>		<b>Enums</b>	
<a href="#">KeyRep.Type</a>	Key type.	<a href="#">KeyRep.Type</a>	<a href="#">Type</a> enumerates the supported key types.
<b>Exception Summary</b>		<b>Exceptions</b>	
<a href="#">AccessControlException</a>	This exception is thrown by the AccessController to indicate that a requested access (to a critical system resource such as the file system or the network) is denied.	<a href="#">AccessControlException</a>	<a href="#">AccessControlException</a> is thrown if the access control infrastructure denies protected access due to missing permissions.
<a href="#">DigestException</a>	This is the generic Message Digest exception.	<a href="#">DigestException</a>	<a href="#">DigestException</a> is a general message digest exception.
<a href="#">GeneralSecurityException</a>	The GeneralSecurityException class is a generic security exception class that provides type safety for all the security-related exception classes that extend from it.	<a href="#">GeneralSecurityException</a>	<a href="#">GeneralSecurityException</a> is a general security exception and the superclass for all security specific exceptions.
<a href="#">InvalidAlgorithmParameterException</a>	This is the exception for invalid or inappropriate algorithm parameters.	<a href="#">InvalidAlgorithmParameterException</a>	<a href="#">InvalidAlgorithmParameterException</a> indicates the occurrence of invalid algorithm parameters.
<a href="#">InvalidKeyException</a>	This is the exception for invalid Keys (invalid encoding, wrong length, uninitialized, etc).	<a href="#">InvalidKeyException</a>	<a href="#">InvalidKeyException</a> indicates exceptional conditions, caused by an invalid key.
<a href="#">InvalidParameterException</a>	This exception, designed for use by the JCA/JCE engine classes, is thrown when		

Java™ 2 Platform Standard Edition 5.0 API Specification (java.security)		Android APIs (java.security)	
<a href="#"><u>KeyException</u></a>	This is the basic key exception.	<a href="#"><u>InvalidParameterException</u></a>	<a href="#"><u>InvalidParameterException</u></a> indicates exceptional conditions, caused by invalid parameters.
<a href="#"><u>KeyManagementException</u></a>	This is the general key management exception for all operations dealing with key management.	<a href="#"><u>KeyException</u></a>	<a href="#"><u>KeyException</u></a> is the common superclass of all key related exceptions.
<a href="#"><u>KeyStoreException</u></a>	This is the generic KeyStore exception.	<a href="#"><u>KeyManagementException</u></a>	<a href="#"><u>KeyManagementException</u></a> is a general exception, thrown to indicate an exception during processing an operation concerning key management.
<a href="#"><u>NoSuchAlgorithmException</u></a>	This exception is thrown when a particular cryptographic algorithm is requested but is not available in the environment.	<a href="#"><u>KeyStoreException</u></a>	<a href="#"><u>KeyStoreException</u></a> is a general <a href="#"><u>KeyStore</u></a> exception.
<a href="#"><u>NoSuchProviderException</u></a>	This exception is thrown when a particular security provider is requested but is not available in the environment.	<a href="#"><u>NoSuchAlgorithmException</u></a>	<a href="#"><u>NoSuchAlgorithmException</u></a> indicates that a requested algorithm could not be found.
<a href="#"><u>PrivilegedActionException</u></a>	This exception is thrown by <code>doPrivileged(PrivilegedExceptionAction)</code> and <code>doPrivileged(PrivilegedExceptionAction, AccessControlContext context)</code> to indicate that the action being performed threw a checked exception.	<a href="#"><u>NoSuchProviderException</u></a>	<a href="#"><u>NoSuchProviderException</u></a> indicates that a requested security provider could not be found.
<a href="#"><u>ProviderException</u></a>	A runtime exception for Provider exceptions (such as misconfiguration errors or unrecoverable internal errors), which may be subclassed by Providers to throw specialized, provider-specific runtime	<a href="#"><u>PrivilegedActionException</u></a>	<a href="#"><u>PrivilegedActionException</u></a> wraps exceptions which are thrown from within privileged operations.

Java™ 2 Platform Standard Edition 5.0 API Specification (java.security)		Android APIs (java.security)	
<a href="#"><u>SignatureException</u></a>	This is the generic Signature exception.	<a href="#"><u>ProviderException</u></a>	<a href="#"><u>ProviderException</u></a> is a general exception, thrown by security <a href="#"><u>Providers</u></a> .
<a href="#"><u>UnrecoverableEntryException</u></a>	This exception is thrown if an entry in the keystore cannot be recovered.	<a href="#"><u>SignatureException</u></a>	<a href="#"><u>SignatureException</u></a> is a general <a href="#"><u>Signature</u></a> exception.
<a href="#"><u>UnrecoverableKeyException</u></a>	This exception is thrown if a key in the keystore cannot be recovered.	<a href="#"><u>UnrecoverableEntryException</u></a>	<a href="#"><u>UnrecoverableEntryException</u></a> indicates, that a <a href="#"><u>KeyStore.Entry</u></a> cannot be recovered from a <a href="#"><u>KeyStore</u></a> .
		<a href="#"><u>UnrecoverableKeyException</u></a>	<a href="#"><u>UnrecoverableKeyException</u></a> indicates, that a key cannot be recovered from a <a href="#"><u>KeyStore</u></a> .

## Exhibit Copyright-E

Java™ 2 Platform Standard Edition 5.0 API Specification ( <a href="#">java.security.KeyPair</a> )	Android APIs ( <a href="#">java.security.KeyPair</a> )
<p><a href="#">java.security</a> <b>Class <a href="#">KeyPair</a></b></p> <p><a href="#">java.lang.Object</a> └─ <a href="#">java.security.KeyPair</a></p> <p><b>All Implemented Interfaces:</b> <a href="#">Serializable</a></p> <hr/> <p>public final class <b>KeyPair</b> extends <a href="#">Object</a> implements <a href="#">Serializable</a></p> <p>This class is a simple holder for a key pair (a public key and a private key). It does not enforce any security, and, when initialized, should be treated like a <a href="#">PrivateKey</a>.</p> <p><b>See Also:</b> <a href="#">PublicKey</a>, <a href="#">PrivateKey</a>, <a href="#">Serialized Form</a></p>	<p>public final class <b>KeyPair</b></p> <p>extends <a href="#">Object</a> implements <a href="#">Serializable</a> <a href="#">java.lang.Object</a> └─ <a href="#">java.security.KeyPair</a></p> <hr/> <p><b>Class Overview</b></p> <p><a href="#">KeyPair</a> is a container for a public key and a private key. Since the private key can be accessed, instances must be treated like a private key.</p> <p><b>See Also</b></p> <ul style="list-style-type: none"> <li>• <a href="#">PrivateKey</a></li> <li>• <a href="#">PublicKey</a></li> </ul>
<p><b>Constructor Summary</b></p> <div style="border: 1px solid black; padding: 5px;"> <p><a href="#">KeyPair</a>(<a href="#">PublicKey</a> publicKey, <a href="#">PrivateKey</a> privateKey) Constructs a key pair from the given public key and private key.</p> </div>	<p><b>Public Constructors</b></p> <div style="border: 1px solid black; padding: 5px;"> <p>public <b>KeyPair</b> (<a href="#">PublicKey</a> publicKey, <a href="#">PrivateKey</a> privateKey)</p> <p>Since: API Level 1</p> <p>Constructs a new instance of <a href="#">KeyPair</a> with a public key and the corresponding private key.</p> <p><b>Parameters</b></p> </div>

	<div><div><div><div><div></div><div><i>publicKey</i></div></div><div><div></div><div>the public key.</div></div></div></div><div><div><div><div></div><div><i>privateKey</i></div></div><div><div></div><div>the private key.</div></div></div></div></div>
<div><div><div><div><div></div><div>Method Summary</div></div></div></div></div>	
<div><div><div><div></div><div><a href="#">PrivateKey</a></div></div></div></div>	<div><div><div><div><div></div><div><a href="#">getPrivate</a></div></div><div><div></div><div>()</div></div></div><div><div></div><div>Returns a reference to the private key component of this key pair.</div></div></div></div>
<div><div><div><div></div><div><a href="#">PublicKey</a></div></div></div></div>	<div><div><div><div><div></div><div><a href="#">getPublic</a></div></div><div><div></div><div>()</div></div></div><div><div></div><div>Returns a reference to the public key component of this key pair.</div></div></div></div>
<div><div><div><div><div></div><div>Methods inherited from class <a href="#">java.lang.Object</a></div></div></div></div></div>	
<div><div><div><div><div></div><div><a href="#">clone</a>, <a href="#">equals</a>, <a href="#">finalize</a>, <a href="#">getClass</a>, <a href="#">hashCode</a>, <a href="#">notify</a>, <a href="#">notifyAll</a>, <a href="#">toString</a>, <a href="#">wait</a>, <a href="#">wait</a>, <a href="#">wait</a></div></div></div></div></div>	

<div><div><div><div><div></div><div>Summary</div></div></div></div></div>	
<div><div><div><div><div></div><div>Public Constructors</div></div></div></div></div>	
<div><div><div><div></div><div><a href="#">KeyPair</a></div></div></div></div>	<div><div><div><div><div></div><div><a href="#">KeyPair</a></div></div><div><div></div><div>(<a href="#">PublicKey</a> publicKey, <a href="#">PrivateKey</a> privateKey)</div></div></div><div><div></div><div>Constructs a new instance of <code>KeyPair</code> with a public key and the corresponding private key.</div></div></div></div>
<div><div><div><div><div></div><div>Public Methods</div></div></div></div></div>	
<div><div><div><div></div><div><a href="#">PrivateKey</a></div></div></div></div>	<div><div><div><div><div></div><div><a href="#">getPrivate</a></div></div><div><div></div><div>()</div></div></div><div><div></div><div>Returns the private key.</div></div></div></div>
<div><div><div><div></div><div><a href="#">PublicKey</a></div></div></div></div>	<div><div><div><div><div></div><div><a href="#">getPublic</a></div></div><div><div></div><div>()</div></div></div><div><div></div><div>Returns the public key.</div></div></div></div>
<div><div><div><div><div></div><div><a href="#">[Expand]</a></div></div></div></div></div>	
<div><div><div><div><div></div><div>Inherited Methods</div></div></div></div></div>	
<div><div><div><div><div></div><div>►From class <a href="#">java.lang.Object</a></div></div></div></div></div>	



<h2>Constructor Detail</h2> <h3>KeyPair</h3> <pre>public KeyPair(<a href="#">PublicKey</a> publicKey,                <a href="#">PrivateKey</a> privateKey)</pre> <p>Constructs a key pair from the given public key and private key.</p> <p>Note that this constructor only stores references to the public and private key components in the generated key pair. This is safe, because <code>Key</code> objects are immutable.</p> <p><b>Parameters:</b>  <code>publicKey</code> - the public key.  <code>privateKey</code> - the private key.</p>	<h2>Public Constructors</h2> <pre>public KeyPair (<a href="#">PublicKey</a> publicKey, <a href="#">PrivateKey</a> privateKey)</pre> <p>Since: API Level 1</p> <p>Constructs a new instance of <code>KeyPair</code> with a public key and the corresponding private key.</p> <p><b>Parameters</b></p> <p><i>publicKey</i>     the public key.</p> <p><i>privateKey</i>    the private key.</p>
<h2>Method Detail</h2> <h3>getPublic</h3> <pre>public <a href="#">PublicKey</a> getPublic()</pre> <p>Returns a reference to the public key component of this key pair.</p> <p><b>Returns:</b>  a reference to the public key.</p> <hr/> <h3>getPrivate</h3> <pre>public <a href="#">PrivateKey</a> getPrivate()</pre> <p>Returns a reference to the private key component of this key pair.</p> <p><b>Returns:</b></p>	<h2>Public Methods</h2> <pre>public <a href="#">PrivateKey</a> getPrivate ()</pre> <p>Since: API Level 1</p> <p>Returns the private key.</p> <p><b>Returns</b></p> <ul style="list-style-type: none"> <li>the private key.</li> </ul> <pre>public <a href="#">PublicKey</a> getPublic ()</pre> <p>Since: API Level 1</p> <p>Returns the public key.</p>

<p>a reference to the private key.</p> <hr/>	<p><b>Returns</b></p> <ul style="list-style-type: none"><li>• the public key.</li></ul>
--	---

## Exhibit Copyright-F

Java™ 2 Platform Standard Edition 5.0 API Specification (java.lang.Runtime)	Android APIs (java.lang.Runtime)								
<p>java lang</p> <h3>Class Runtime</h3> <p><a href="#">java.lang.Object</a></p> <p>└─ <b>java.lang.Runtime</b></p> <hr/> <p>public class <b>Runtime</b> extends <a href="#">Object</a></p> <p>Every Java application has a single instance of class <code>Runtime</code> that allows the application to interface with the environment in which the application is running. The current runtime can be obtained from the <code>getRuntime</code> method.</p> <p>An application cannot create its own instance of this class.</p> <p><b>Since:</b> JDK1.0</p> <p><b>See Also:</b> <a href="#">getRuntime()</a></p>	<p>public class</p> <h3>Runtime</h3> <p>extends <a href="#">Object</a></p> <p><a href="#">java.lang.Object</a></p> <p>↳ java.lang.Runtime</p> <hr/> <h3>Class Overview</h3> <p>Allows Java applications to interface with the environment in which they are running. Applications can not create an instance of this class, but they can get a singleton instance by invoking <a href="#">getRuntime()</a>.</p> <p><b>See Also</b></p> <p><a href="#">System</a></p>								
<h3>Method Summary</h3> <table border="1"> <tr> <td>void</td><td><a href="#">addShutdownHook</a>(<a href="#">Thread</a> hook) Registers a new virtual-machine shutdown hook.</td></tr> <tr> <td>int</td><td><a href="#">availableProcessors</a>() Returns the number of processors available to the Java virtual machine.</td></tr> </table>	void	<a href="#">addShutdownHook</a> ( <a href="#">Thread</a> hook) Registers a new virtual-machine shutdown hook.	int	<a href="#">availableProcessors</a> () Returns the number of processors available to the Java virtual machine.	<h3>Summary</h3> <p><b>Public Methods</b></p> <table border="1"> <tr> <td>void</td><td><a href="#">addShutdownHook</a>(<a href="#">Thread</a> hook) Registers a virtual-machine shutdown hook.</td></tr> <tr> <td>int</td><td><a href="#">availableProcessors</a>() Returns the number of processors available to the virtual machine.</td></tr> </table>	void	<a href="#">addShutdownHook</a> ( <a href="#">Thread</a> hook) Registers a virtual-machine shutdown hook.	int	<a href="#">availableProcessors</a> () Returns the number of processors available to the virtual machine.
void	<a href="#">addShutdownHook</a> ( <a href="#">Thread</a> hook) Registers a new virtual-machine shutdown hook.								
int	<a href="#">availableProcessors</a> () Returns the number of processors available to the Java virtual machine.								
void	<a href="#">addShutdownHook</a> ( <a href="#">Thread</a> hook) Registers a virtual-machine shutdown hook.								
int	<a href="#">availableProcessors</a> () Returns the number of processors available to the virtual machine.								

Java™ 2 Platform Standard Edition 5.0 API Specification (java.lang.Runtime)		Android APIs (java.lang.Runtime)	
<a href="#">Process</a>	<a href="#">exec</a> ( <a href="#">String</a> command) Executes the specified string command in a separate process.	<a href="#">Process</a>	<a href="#">exec</a> ( <a href="#">String</a> [] progArray, <a href="#">String</a> [] envp) Executes the specified command and its arguments in a separate native process.
<a href="#">Process</a>	<a href="#">exec</a> ( <a href="#">String</a> [] cmdarray) Executes the specified command and arguments in a separate process.	<a href="#">Process</a>	<a href="#">exec</a> ( <a href="#">String</a> prog, <a href="#">String</a> [] envp, <a href="#">File</a> directory) Executes the specified program in a separate native process.
<a href="#">Process</a>	<a href="#">exec</a> ( <a href="#">String</a> [] cmdarray, <a href="#">String</a> [] envp) Executes the specified command and arguments in a separate process with the specified environment.	<a href="#">Process</a>	<a href="#">exec</a> ( <a href="#">String</a> [] progArray, <a href="#">String</a> [] envp, <a href="#">File</a> directory) Executes the specified command and its arguments in a separate native process.
<a href="#">Process</a>	<a href="#">exec</a> ( <a href="#">String</a> [] cmdarray, <a href="#">String</a> [] envp, <a href="#">File</a> dir) Executes the specified command and arguments in a separate process with the specified environment and working directory.	<a href="#">Process</a>	<a href="#">exec</a> ( <a href="#">String</a> prog, <a href="#">String</a> [] envp) Executes the specified program in a separate native process.
<a href="#">Process</a>	<a href="#">exec</a> ( <a href="#">String</a> command, <a href="#">String</a> [] envp) Executes the specified string command in a separate process with the specified environment.	<a href="#">Process</a>	<a href="#">exec</a> ( <a href="#">String</a> prog) Executes the specified program in a separate native process.
<a href="#">Process</a>	<a href="#">exec</a> ( <a href="#">String</a> command, <a href="#">String</a> [] envp, <a href="#">File</a> dir) Executes the specified string command in a separate process with the specified environment and working directory.	<a href="#">Process</a>	<a href="#">exec</a> ( <a href="#">String</a> [] progArray) Executes the specified command and its arguments in a separate native process.
void	<a href="#">exit</a> (int status) Terminates the currently running Java virtual machine by initiating its shutdown sequence.	void	<a href="#">exit</a> (int code) Causes the virtual machine to stop running and the program to exit.

Java™ 2 Platform Standard Edition 5.0 API Specification (java.lang.Runtime)		Android APIs (java.lang.Runtime)	
long	<a href="#">freeMemory</a> () Returns the amount of free memory in the Java Virtual Machine.	long	<a href="#">freeMemory</a> () Returns the amount of free memory resources which are available to the running program.
void	<a href="#">gc</a> () Runs the garbage collector.	void	<a href="#">gc</a> () Indicates to the virtual machine that it would be a good time to run the garbage collector.
<a href="#">InputStream</a>	<a href="#">getLocalizedInputStream</a> ( <a href="#">InputStream</a> in) <b>Deprecated.</b> As of JDK 1.1, the preferred way to translate a byte stream in the local encoding into a character stream in Unicode is via the <i>InputStreamReader</i> and <i>BufferedReader</i> classes.	<a href="#">InputStream</a>	<a href="#">getLocalizedInputStream</a> ( <a href="#">InputStream</a> stream) <i>This method is deprecated. Use <a href="#">InputStreamReader</a>.</i>
<a href="#">OutputStream</a>	<a href="#">getLocalizedOutputStream</a> ( <a href="#">OutputStream</a> out) <b>Deprecated.</b> As of JDK 1.1, the preferred way to translate a Unicode character stream into a byte stream in the local encoding is via the <i>OutputStreamWriter</i> , <i>BufferedWriter</i> , and <i>PrintWriter</i> classes.	<a href="#">OutputStream</a>	<a href="#">getLocalizedOutputStream</a> ( <a href="#">OutputStream</a> stream) <i>This method is deprecated. Use <a href="#">OutputStreamWriter</a>.</i>
static <a href="#">Runtime</a>	<a href="#">getRuntime</a> () Returns the runtime object associated with the current Java application.	static <a href="#">Runtime</a>	<a href="#">getRuntime</a> () Returns the single <a href="#">Runtime</a> instance.
void	<a href="#">halt</a> (int status) Forcibly terminates the currently running Java virtual machine.	void	<a href="#">halt</a> (int code) Causes the virtual machine to stop running, and the program to exit.
void	<a href="#">load</a> ( <a href="#">String</a> filename) Loads the specified filename as a dynamic library.	void	<a href="#">load</a> ( <a href="#">String</a> pathName) Loads and links the dynamic library that is identified through the specified path.
void	<a href="#">loadLibrary</a> ( <a href="#">String</a> libname) Loads the dynamic library with the specified library name.	void	<a href="#">loadLibrary</a> ( <a href="#">String</a> libName) Loads and links the library with the specified name.
		long	<a href="#">maxMemory</a> ()

Java™ 2 Platform Standard Edition 5.0 API Specification (java.lang.Runtime)		Android APIs (java.lang.Runtime)	
long	<a href="#">maxMemory()</a> Returns the maximum amount of memory that the Java virtual machine will attempt to use.		Returns the maximum amount of memory that may be used by the virtual machine, or <code>Long.MAX_VALUE</code> if there is no such limit.
boolean	<a href="#">removeShutdownHook(Thread hook)</a> De-registers a previously-registered virtual-machine shutdown hook.	boolean	<a href="#">removeShutdownHook(Thread hook)</a> Unregisters a previously registered virtual machine shutdown hook.
void	<a href="#">runFinalization()</a> Runs the finalization methods of any objects pending finalization.	void	<a href="#">runFinalization()</a> Provides a hint to the virtual machine that it would be useful to attempt to perform any outstanding object finalization.
static void	<a href="#">runFinalizersOnExit(boolean value)</a> <b>Deprecated.</b> <i>This method is inherently unsafe. It may result in finalizers being called on live objects while other threads are concurrently manipulating those objects, resulting in erratic behavior or deadlock.</i>	static void	<a href="#">runFinalizersOnExit(boolean run)</a> <i>This method is deprecated. This method is unsafe.</i>
long	<a href="#">totalMemory()</a> Returns the total amount of memory in the Java virtual machine.	long	<a href="#">totalMemory()</a> Returns the total amount of memory which is available to the running program.
void	<a href="#">traceInstructions(boolean on)</a> Enables/Disables tracing of instructions.	void	<a href="#">traceInstructions(boolean enable)</a> Switches the output of debug information for instructions on or off.
void	<a href="#">traceMethodCalls(boolean on)</a> Enables/Disables tracing of method calls.	void	<a href="#">traceMethodCalls(boolean enable)</a> Switches the output of debug information for methods on or off.
<b>Methods inherited from class java.lang.Object</b> <a href="#">clone</a> , <a href="#">equals</a> , <a href="#">finalize</a> , <a href="#">getClass</a> , <a href="#">hashCode</a> , <a href="#">notify</a> , <a href="#">notifyAll</a> , <a href="#">toString</a> , <a href="#">wait</a> , <a href="#">wait</a> , <a href="#">wait</a>		<b>Inherited Methods<sup>[1]</sup></b>	

<sup>1</sup> Collapsed view.

Java™ 2 Platform Standard Edition 5.0 API Specification (java.lang.Runtime)	Android APIs (java.lang.Runtime)
	<div>►From class <a href="#">java.lang.Object</a></div>
	<div>Inherited Methods<sup>[2]</sup></div> <div>▼From class <a href="#">java.lang.Object</a></div> <div><div><div>Object</div><div>clone()</div><div>Creates and returns a copy of this <a href="#">Object</a>.</div></div><div><div>boolean</div><div><a href="#">equals(Object o)</a></div><div>Compares this instance with the specified object and indicates if they are equal.</div></div><div><div>void</div><div><a href="#">finalize()</a></div><div>Called before the object's memory is reclaimed by the VM.</div></div><div><div>final <a href="#">Class&lt;? extends Object&gt;</a></div><div><a href="#">getClass()</a></div><div>Returns the unique instance of <a href="#">Class</a> that represents this object's class.</div></div><div><div>int</div><div><a href="#">hashCode()</a></div><div>Returns an integer hash code for this object.</div></div><div><div>final void</div><div><a href="#">notify()</a></div><div>Causes a thread which is waiting on this object's monitor (by means of calling one of the <a href="#">wait()</a> methods) to be woken up.</div></div><div><div>final void</div><div><a href="#">notifyAll()</a></div><div>Causes all threads which are waiting on this object's monitor (by means of calling one of the</div></div></div>

Java™ 2 Platform Standard Edition 5.0 API Specification (java.lang.Runtime)	Android APIs (java.lang.Runtime)
	<div><div></div><div><div>wait() methods) to be woken up.</div><div>StringtoString() Returns a string containing a concise, human-readable description of this object.</div><div>final voidwait() Causes the calling thread to wait until another thread calls the notify() or notifyAll() method of this object.</div><div>final voidwait(long millis, int nanos) Causes the calling thread to wait until another thread calls the notify() or notifyAll() method of this object or until the specified timeout expires.</div><div>final voidwait(long millis) Causes the calling thread to wait until another thread calls the notify() or notifyAll() method of this object or until the specified timeout expires.</div></div></div>



Java™ 2 Platform Standard Edition 5.0 API Specification (java.lang.Runtime)	Android APIs (java.lang.Runtime)
<b>Method Detail</b>	<b>Public Methods</b>
<p><b>addShutdownHook</b></p> <pre>public void addShutdownHook(<a href="#">Thread</a> hook)</pre> <p>Registers a new virtual-machine shutdown hook.</p> <p>The Java virtual machine <i>shuts down</i> in response to two kinds of events:</p> <ul style="list-style-type: none"> <li>• The program <i>exits</i> normally, when the last non-daemon thread exits or when the <a href="#">exit</a> (equivalently, <a href="#">System.exit</a>) method is invoked, or</li> <li>• The virtual machine is <i>terminated</i> in response to a user interrupt, such as typing ^C, or a system-wide event, such as user logoff or system shutdown.</li> </ul> <p>A <i>shutdown hook</i> is simply an initialized but unstarted thread. When the virtual machine begins its shutdown sequence it will start all registered shutdown hooks in some unspecified order and let them run concurrently. When all the hooks have finished it will then run all uninvoked finalizers if finalization-on-exit has been enabled. Finally, the virtual machine will halt. Note that daemon threads will continue to run during the shutdown sequence, as will non-daemon threads if shutdown was initiated by invoking the <a href="#">exit</a> method.</p> <p>Once the shutdown sequence has begun it can be stopped only by invoking the <a href="#">halt</a> method, which forcibly terminates the virtual machine.</p> <p>Once the shutdown sequence has begun it is impossible to register a new shutdown hook or de-register a previously-registered hook. Attempting either of these operations will cause an <a href="#">IllegalStateException</a> to be thrown.</p>	<pre>public void addShutdownHook(<a href="#">Thread</a> hook)</pre> <p>Since: API Level 1</p> <p>Registers a virtual-machine shutdown hook. A shutdown hook is a <a href="#">Thread</a> that is ready to run, but has not yet been started. All registered shutdown hooks will be executed once the virtual machine shuts down properly. A proper shutdown happens when either the <a href="#">exit(int)</a> method is called or the surrounding system decides to terminate the application, for example in response to a CTRL-C or a system-wide shutdown. A termination of the virtual machine due to the <a href="#">halt(int)</a> method, an <a href="#">Error</a> or a SIGKILL, in contrast, is not considered a proper shutdown. In these cases the shutdown hooks will not be run.</p> <p>Shutdown hooks are run concurrently and in an unspecified order. Hooks failing due to an unhandled exception are not a problem, but the stack trace might be printed to the console. Once initiated, the whole shutdown process can only be terminated by calling <code>halt()</code>.</p> <p>If <a href="#">runFinalizersOnExit(boolean)</a> has been called with a <code>true</code> argument, garbage collection and finalization will take place after all hooks are either finished or have failed. Then the virtual machine terminates.</p> <p>It is recommended that shutdown hooks do not do any time-consuming activities, in order to not hold up the shutdown process longer than necessary.</p> <p><b>Parameters</b></p> <p><code>hook</code> the shutdown hook to register.</p> <p><b>Throws</b></p> <p><a href="#">IllegalArgumentExeption</a> if the hook has already been started or if it has already been registered.</p> <p><a href="#">IllegalStateException</a> if the virtual machine is already shutting</p>

<b>Java™ 2 Platform Standard Edition 5.0 API Specification</b> <b>(java.lang.Runtime)</b>	<b>Android APIs</b> <b>(java.lang.Runtime)</b>
<p>Shutdown hooks run at a delicate time in the life cycle of a virtual machine and should therefore be coded defensively. They should, in particular, be written to be thread-safe and to avoid deadlocks insofar as possible. They should also not rely blindly upon services that may have registered their own shutdown hooks and therefore may themselves in the process of shutting down.</p> <p>Shutdown hooks should also finish their work quickly. When a program invokes <a href="#">exit</a> the expectation is that the virtual machine will promptly shut down and exit. When the virtual machine is terminated due to user logoff or system shutdown the underlying operating system may only allow a fixed amount of time in which to shut down and exit. It is therefore inadvisable to attempt any user interaction or to perform a long-running computation in a shutdown hook.</p> <p>Uncaught exceptions are handled in shutdown hooks just as in any other thread, by invoking the <a href="#">uncaughtException</a> method of the thread's <a href="#">ThreadGroup</a> object. The default implementation of this method prints the exception's stack trace to <a href="#">System.err</a> and terminates the thread; it does not cause the virtual machine to exit or halt.</p> <p>In rare circumstances the virtual machine may <i>abort</i>, that is, stop running without shutting down cleanly. This occurs when the virtual machine is terminated externally, for example with the SIGKILL signal on Unix or the TerminateProcess call on Microsoft Windows. The virtual machine may also abort if a native method goes awry by, for example, corrupting internal data structures or attempting to access nonexistent memory. If the virtual machine aborts then no guarantee can be made about whether or not any shutdown hooks will be run.</p> <p><b>Parameters:</b>  hook - An initialized but unstarted <a href="#">Thread</a> object</p> <p><b>Throws:</b></p>	<p>down.</p> <p><a href="#">SecurityException</a></p> <p>if a SecurityManager is registered and the calling code doesn't have the RuntimePermission("shutdownHooks").</p>

<b>Java™ 2 Platform Standard Edition 5.0 API Specification</b> <b>(java.lang.Runtime)</b>	<b>Android APIs</b> <b>(java.lang.Runtime)</b>
<p><a href="#">IllegalArgumentException</a> - If the specified hook has already been registered, or if it can be determined that the hook is already running or has already been run</p> <p><a href="#">IllegalStateException</a> - If the virtual machine is already in the process of shutting down</p> <p><a href="#">SecurityException</a> - If a security manager is present and it denies <a href="#">RuntimePermission</a>( "shutdownHooks" )</p> <p><b>Since:</b> 1.3</p> <p><b>See Also:</b> <a href="#">removeShutdownHook( java.lang.Thread)</a>, <a href="#">halt(int)</a>, <a href="#">exit(int)</a></p>	
<p><b>availableProcessors</b></p> <p>public int <b>availableProcessors</b>()</p> <p>Returns the number of processors available to the Java virtual machine.</p> <p>This value may change during a particular invocation of the virtual machine. Applications that are sensitive to the number of available processors should therefore occasionally poll this property and adjust their resource usage appropriately.</p> <p><b>Returns:</b> the maximum number of processors available to the virtual machine; never smaller than one</p> <p><b>Since:</b> 1.4</p>	<p>public int <b>availableProcessors</b> ()</p> <p>Since: API Level 1</p> <p>Returns the number of processors available to the virtual machine.</p> <p><b>Returns</b> the number of available processors, at least 1.</p>

Java™ 2 Platform Standard Edition 5.0 API Specification (java.lang.Runtime)	Android APIs (java.lang.Runtime)
<p><b>exit</b></p> <pre>public void <b>exit</b>(int status)</pre> <p>Terminates the currently running Java virtual machine by initiating its shutdown sequence. This method never returns normally. The argument serves as a status code; by convention, a nonzero status code indicates abnormal termination.</p> <p>The virtual machine's shutdown sequence consists of two phases. In the first phase all registered <a href="#">shutdown hooks</a>, if any, are started in some unspecified order and allowed to run concurrently until they finish. In the second phase all uninvoked finalizers are run if <a href="#">finalization-on-exit</a> has been enabled. Once this is done the virtual machine <a href="#">halts</a>.</p> <p>If this method is invoked after the virtual machine has begun its shutdown sequence then if shutdown hooks are being run this method will block indefinitely. If shutdown hooks have already been run and on-exit finalization has been enabled then this method halts the virtual machine with the given status code if the status is nonzero; otherwise, it blocks indefinitely.</p> <p>The <a href="#">System.exit</a> method is the conventional and convenient means of invoking this method.</p> <p><b>Parameters:</b>  status - Termination status. By convention, a nonzero status code indicates abnormal termination.</p> <p><b>Throws:</b>  <a href="#">SecurityException</a> - If a security manager is present and its <a href="#">checkExit</a> method does not permit exiting with the specified status</p> <p><b>See Also:</b>  <a href="#">SecurityException</a>, <a href="#">SecurityManager.checkExit(int)</a>,  <a href="#">addShutdownHook(java.lang.Thread)</a>,  <a href="#">removeShutdownHook(java.lang.Thread)</a>,</p>	<pre>public <a href="#">Process</a> <b>exec</b> (<a href="#">String</a>[] progArray, <a href="#">String</a>[] envp)</pre> <p>Since: API Level 1</p> <p>Executes the specified command and its arguments in a separate native process. The new process uses the environment provided in envp. Calling this method is equivalent to calling <code>exec(progArray, envp, null)</code>.</p> <p><b>Parameters</b></p> <p><i>progArray</i>      the array containing the program to execute as well as any arguments to the program.</p> <p><i>envp</i>            the array containing the environment to start the new process in.</p> <p><b>Returns</b></p> <p>the new <code>Process</code> object that represents the native process.</p> <p><b>Throws</b></p> <p><a href="#">IOException</a>      if the requested program can not be executed.</p> <p><a href="#">SecurityException</a>    if the current <code>SecurityManager</code> disallows program execution.</p> <p><b>See Also</b></p> <p><a href="#">checkExec(String)</a></p> <pre>public <a href="#">Process</a> <b>exec</b> (<a href="#">String</a> prog, <a href="#">String</a>[] envp, <a href="#">File</a> directory)</pre> <p>Since: API Level 1</p> <p>Executes the specified program in a separate native process. The new process uses the environment provided in envp and the working directory specified by directory.</p> <p><b>Parameters</b></p>

<b>Java™ 2 Platform Standard Edition 5.0 API Specification</b> <b>(java.lang.Runtime)</b>	<b>Android APIs</b> <b>(java.lang.Runtime)</b>
<p><a href="#">runFinalizersOnExit(boolean)</a>, <a href="#">halt(int)</a></p> <p>...</p> <hr/> <p><b>exec</b></p> <p>public <a href="#">Process</a> <b>exec</b>(<a href="#">String</a> command)  throws <a href="#">IOException</a></p> <p>Executes the specified string command in a separate process.</p> <p>This is a convenience method. An invocation of the form <code>exec(command)</code> behaves in exactly the same way as the invocation <code>exec(command, null, null)</code>.</p> <p><b>Parameters:</b>  command - a specified system command.</p> <p><b>Returns:</b>  A new <a href="#">Process</a> object for managing the subprocess</p> <p><b>Throws:</b>  <a href="#">SecurityException</a> - If a security manager exists and its <a href="#">checkExec</a> method doesn't allow creation of the subprocess  <a href="#">IOException</a> - If an I/O error occurs  <a href="#">NullPointerException</a> - If command is null  <a href="#">IllegalArgumentException</a> - If command is empty</p> <p><b>See Also:</b>  <a href="#">exec(String[], String[], File)</a>, <a href="#">ProcessBuilder</a></p>	<p><i>prog</i> the name of the program to execute.</p> <p><i>envp</i> the array containing the environment to start the new process in.</p> <p><i>directory</i> the directory in which to execute the program. If null, execute if in the same directory as the parent process.</p> <p><b>Returns</b>  the new <code>Process</code> object that represents the native process.</p> <p><b>Throws</b>  <a href="#">IOException</a> if the requested program can not be executed.  <a href="#">SecurityException</a> if the current <code>SecurityManager</code> disallows program execution.</p> <p><b>See Also</b>  <a href="#">checkExec(String)</a></p> <hr/> <p>public <a href="#">Process</a> <b>exec</b> (<a href="#">String[]</a> progArray, <a href="#">String[]</a> envp, <a href="#">File</a> directory)</p> <p>Since: API Level 1</p> <p>Executes the specified command and its arguments in a separate native process. The new process uses the environment provided in <code>envp</code> and the working directory specified by <code>directory</code>.</p> <p><b>Parameters</b></p> <p><i>progArray</i> the array containing the program to execute as well as any arguments to the program.</p> <p><i>envp</i> the array containing the environment to start the new process in.</p>
<p><b>exec</b></p> <p>public <a href="#">Process</a> <b>exec</b>(<a href="#">String</a> command,  <a href="#">String[]</a> envp)  throws <a href="#">IOException</a></p> <p>Executes the specified string command in a separate process with the specified environment.</p>	

Java™ 2 Platform Standard Edition 5.0 API Specification (java.lang.Runtime)	Android APIs (java.lang.Runtime)
<p>This is a convenience method. An invocation of the form <code>exec(command, envp)</code> behaves in exactly the same way as the invocation <code>exec(command, envp, null)</code>.</p> <p><b>Parameters:</b></p> <p><code>command</code> - a specified system command.</p> <p><code>envp</code> - array of strings, each element of which has environment variable settings in the format <i>name=value</i>, or <code>null</code> if the subprocess should inherit the environment of the current process.</p> <p><b>Returns:</b></p> <p>A new <a href="#">Process</a> object for managing the subprocess</p> <p><b>Throws:</b></p> <p><a href="#">SecurityException</a> - If a security manager exists and its <a href="#">checkExec</a> method doesn't allow creation of the subprocess</p> <p><a href="#">IOException</a> - If an I/O error occurs</p> <p><a href="#">NullPointerException</a> - If <code>command</code> is <code>null</code>, or one of the elements of <code>envp</code> is <code>null</code></p> <p><a href="#">IllegalArgumentException</a> - If <code>command</code> is empty</p> <p><b>See Also:</b></p> <p><a href="#">exec(String[], String[], File)</a>, <a href="#">ProcessBuilder</a></p>	<p><i>directory</i> the directory in which to execute the program. If <code>null</code>, execute if in the same directory as the parent process.</p> <p><b>Returns</b></p> <p>the new <code>Process</code> object that represents the native process.</p> <p><b>Throws</b></p> <p><a href="#">IOException</a> if the requested program can not be executed.</p> <p><a href="#">SecurityException</a> if the current <code>SecurityManager</code> disallows program execution.</p> <p><b>See Also</b></p> <p><a href="#">checkExec(String)</a></p>
<p><b>exec</b></p> <pre>public <a href="#">Process</a> <b>exec</b>(<a href="#">String</a> command,                      <a href="#">String</a>[] envp,                      <a href="#">File</a> dir)     throws <a href="#">IOException</a></pre> <p>Executes the specified string command in a separate process with the specified environment and working directory.</p> <p>This is a convenience method. An invocation of the form <code>exec(command, envp, dir)</code> behaves in exactly the same way as the invocation <code>exec(cmdarray, envp, dir)</code>, where <code>cmdarray</code> is an array</p>	<pre>public <a href="#">Process</a> <b>exec</b> (<a href="#">String</a> prog, <a href="#">String</a>[] envp)</pre> <p>Since: API Level 1</p> <p>Executes the specified program in a separate native process. The new process uses the environment provided in <code>envp</code>. Calling this method is equivalent to calling <code>exec(prog, envp, null)</code>.</p> <p><b>Parameters</b></p> <p><i>prog</i> the name of the program to execute.</p> <p><i>envp</i> the array containing the environment to start the new process in.</p> <p><b>Returns</b></p> <p>the new <code>Process</code> object that represents the native process.</p> <p><b>Throws</b></p> <p><a href="#">IOException</a> if the requested program can not be executed.</p>

Java™ 2 Platform Standard Edition 5.0 API Specification (java.lang.Runtime)	Android APIs (java.lang.Runtime)
<p>of all the tokens in <code>command</code>.</p> <p>More precisely, the <code>command</code> string is broken into tokens using a <a href="#">StringTokenizer</a> created by the call <code>new StringTokenizer(command)</code> with no further modification of the character categories. The tokens produced by the tokenizer are then placed in the new string array <code>cmdarray</code>, in the same order.</p> <p><b>Parameters:</b></p> <p><code>command</code> - a specified system command.</p> <p><code>envp</code> - array of strings, each element of which has environment variable settings in the format <i>name=value</i>, or <code>null</code> if the subprocess should inherit the environment of the current process.</p> <p><code>dir</code> - the working directory of the subprocess, or <code>null</code> if the subprocess should inherit the working directory of the current process.</p> <p><b>Returns:</b></p> <p>A new <a href="#">Process</a> object for managing the subprocess</p> <p><b>Throws:</b></p> <p><a href="#">SecurityException</a> - If a security manager exists and its <a href="#">checkExec</a> method doesn't allow creation of the subprocess</p> <p><a href="#">IOException</a> - If an I/O error occurs</p> <p><a href="#">NullPointerException</a> - If <code>command</code> is <code>null</code>, or one of the elements of <code>envp</code> is <code>null</code></p> <p><a href="#">IllegalArgumentException</a> - If <code>command</code> is empty</p> <p><b>Since:</b></p> <p>1.3</p> <p><b>See Also:</b></p> <p><a href="#">ProcessBuilder</a></p>	<p><a href="#">SecurityException</a> if the current <code>SecurityManager</code> disallows program execution.</p> <p><b>See Also</b></p> <p><a href="#">checkExec(String)</a></p> <p>public <a href="#">Process</a> <b>exec</b> (<a href="#">String</a> prog)</p> <p>Since: API Level 1</p> <p>Executes the specified program in a separate native process. The new process inherits the environment of the caller. Calling this method is equivalent to calling <code>exec(prog, null, null)</code>.</p> <p><b>Parameters</b></p> <p><code>prog</code> the name of the program to execute.</p> <p><b>Returns</b></p> <p>the new <code>Process</code> object that represents the native process.</p> <p><b>Throws</b></p> <p><a href="#">IOException</a> if the requested program can not be executed.</p> <p><a href="#">SecurityException</a> if the current <code>SecurityManager</code> disallows program execution.</p> <p><b>See Also</b></p> <p><a href="#">checkExec(String)</a></p> <p>public <a href="#">Process</a> <b>exec</b> (<a href="#">String[]</a> progArray)</p> <p>Since: API Level 1</p> <p>Executes the specified command and its arguments in a separate native process. The new process inherits the environment of the caller. Calling this method is equivalent to calling <code>exec(progArray, null, null)</code>.</p>
<p><b>exec</b></p> <p>public <a href="#">Process</a> <b>exec</b>(<a href="#">String</a>[] cmdarray) throws <a href="#">IOException</a></p> <p>Executes the specified command and arguments in a separate process.</p>	



Java™ 2 Platform Standard Edition 5.0 API Specification (java.lang.Runtime)	Android APIs (java.lang.Runtime)
<p>This is a convenience method. An invocation of the form <code>exec(cmdarray)</code> behaves in exactly the same way as the invocation <code>exec(cmdarray, null, null)</code>.</p> <p><b>Parameters:</b>  <code>cmdarray</code> - array containing the command to call and its arguments.</p> <p><b>Returns:</b>  A new <a href="#">Process</a> object for managing the subprocess</p> <p><b>Throws:</b>  <a href="#">SecurityException</a> - If a security manager exists and its <a href="#">checkExec</a> method doesn't allow creation of the subprocess  <a href="#">IOException</a> - If an I/O error occurs  <a href="#">NullPointerException</a> - If <code>cmdarray</code> is null, or one of the elements of <code>cmdarray</code> is null  <a href="#">IndexOutOfBoundsException</a> - If <code>cmdarray</code> is an empty array (has length 0)</p> <p><b>See Also:</b>  <a href="#">ProcessBuilder</a></p>	<p><b>Parameters</b></p> <p><i>progArray</i> the array containing the program to execute as well as any arguments to the program.</p> <p><b>Returns</b></p> <p>the new <code>Process</code> object that represents the native process.</p> <p><b>Throws</b></p> <p><a href="#">IOException</a> if the requested program can not be executed.</p> <p><a href="#">SecurityException</a> if the current <code>SecurityManager</code> disallows program execution.</p> <p><b>See Also</b></p> <p><a href="#">checkExec(String)</a></p>
<p><b>exec</b></p> <pre>public <a href="#">Process</a> <b>exec</b>(<a href="#">String</a>[] cmdarray,                      <a href="#">String</a>[] envp)     throws <a href="#">IOException</a></pre> <p>Executes the specified command and arguments in a separate process with the specified environment.</p> <p>This is a convenience method. An invocation of the form <code>exec(cmdarray, envp)</code> behaves in exactly the same way as the invocation <code>exec(cmdarray, envp, null)</code>.</p> <p><b>Parameters:</b>  <code>cmdarray</code> - array containing the command to call and its arguments.</p>	<pre>public void <b>exit</b> (int code)</pre> <p>Since: API Level 1</p> <p>Causes the virtual machine to stop running and the program to exit. If <a href="#">runFinalizersOnExit(boolean)</a> has been previously invoked with a <code>true</code> argument, then all objects will be properly garbage-collected and finalized first.</p> <p><b>Parameters</b></p> <p><i>code</i> the return code. By convention, non-zero return codes indicate abnormal terminations.</p> <p><b>Throws</b></p> <p><a href="#">SecurityException</a> if the current <code>SecurityManager</code> does not allow the running thread to terminate the virtual machine.</p> <p><b>See Also</b></p>



<b>Java™ 2 Platform Standard Edition 5.0 API Specification</b> <b>(java.lang.Runtime)</b>	<b>Android APIs</b> <b>(java.lang.Runtime)</b>
<p>envp - array of strings, each element of which has environment variable settings in the format <i>name=value</i>, or null if the subprocess should inherit the environment of the current process.</p> <p><b>Returns:</b> A new <a href="#">Process</a> object for managing the subprocess</p> <p><b>Throws:</b>  <a href="#">SecurityException</a> - If a security manager exists and its <a href="#">checkExec</a> method doesn't allow creation of the subprocess  <a href="#">IOException</a> - If an I/O error occurs  <a href="#">NullPointerException</a> - If cmdarray is null, or one of the elements of cmdarray is null, or one of the elements of envp is null  <a href="#">IndexOutOfBoundsException</a> - If cmdarray is an empty array (has length 0)</p> <p><b>See Also:</b>  <a href="#">ProcessBuilder</a></p> <hr/> <p><b>exec</b></p> <pre>public <a href="#">Process</a> <b>exec</b>(<a href="#">String</a>[] cmdarray,                      <a href="#">String</a>[] envp,                      <a href="#">File</a> dir)     throws <a href="#">IOException</a></pre> <p>Executes the specified command and arguments in a separate process with the specified environment and working directory.</p> <p>Given an array of strings cmdarray, representing the tokens of a command line, and an array of strings envp, representing "environment" variable settings, this method creates a new process in which to execute the specified command.</p> <p>This method checks that cmdarray is a valid operating system command. Which commands are valid is system-dependent, but at the very least the command must be a non-empty list of non-null strings.</p> <p>If envp is null, the subprocess inherits the environment settings of the</p>	<p><a href="#">checkExit(int)</a></p>

Java™ 2 Platform Standard Edition 5.0 API Specification (java.lang.Runtime)	Android APIs (java.lang.Runtime)
<p>current process.</p> <p><a href="#">ProcessBuilder.start()</a> is now the preferred way to start a process with a modified environment.</p> <p>The working directory of the new subprocess is specified by <code>dir</code>. If <code>dir</code> is <code>null</code>, the subprocess inherits the current working directory of the current process.</p> <p>If a security manager exists, its <a href="#">checkExec</a> method is invoked with the first component of the array <code>cmdarray</code> as its argument. This may result in a <a href="#">SecurityException</a> being thrown.</p> <p>Starting an operating system process is highly system-dependent. Among the many things that can go wrong are:</p> <ul style="list-style-type: none"> <li>• The operating system program file was not found.</li> <li>• Access to the program file was denied.</li> <li>• The working directory does not exist.</li> </ul> <p>In such cases an exception will be thrown. The exact nature of the exception is system-dependent, but it will always be a subclass of <a href="#">IOException</a>.</p> <p><b>Parameters:</b>  <code>cmdarray</code> - array containing the command to call and its arguments.  <code>envp</code> - array of strings, each element of which has environment variable settings in the format <i>name=value</i>, or <code>null</code> if the subprocess should inherit the environment of the current process.  <code>dir</code> - the working directory of the subprocess, or <code>null</code> if the subprocess should inherit the working directory of the current process.</p> <p><b>Returns:</b>  A new <a href="#">Process</a> object for managing the subprocess</p> <p><b>Throws:</b></p>	

<b>Java™ 2 Platform Standard Edition 5.0 API Specification</b> <b>(java.lang.Runtime)</b>	<b>Android APIs</b> <b>(java.lang.Runtime)</b>
<p><a href="#">SecurityException</a> - If a security manager exists and its <a href="#">checkExec</a> method doesn't allow creation of the subprocess</p> <p><a href="#">IOException</a> - If an I/O error occurs</p> <p><a href="#">NullPointerException</a> - If cmdarray is null, or one of the elements of cmdarray is null, or one of the elements of envp is null</p> <p><a href="#">IndexOutOfBoundsException</a> - If cmdarray is an empty array (has length 0)</p> <p><b>Since:</b> 1.3</p> <p><b>See Also:</b> <a href="#">ProcessBuilder</a></p>	

Java™ 2 Platform Standard Edition 5.0 API Specification (java.lang.Runtime)	Android APIs (java.lang.Runtime)
<p><b>freeMemory</b></p> <p>public long <b>freeMemory()</b></p> <p>Returns the amount of free memory in the Java Virtual Machine. Calling the <code>gc</code> method may result in increasing the value returned by <code>freeMemory</code>.</p> <p><b>Returns:</b> an approximation to the total amount of memory currently available for future allocated objects, measured in bytes.</p>	<p>public long <b>freeMemory ()</b></p> <p>Since: API Level 1</p> <p>Returns the amount of free memory resources which are available to the running program.</p> <p><b>Returns</b> the approximate amount of free memory, measured in bytes.</p>
<p><b>gc</b></p> <p>public void <b>gc()</b></p> <p>Runs the garbage collector. Calling this method suggests that the Java virtual machine expend effort toward recycling unused objects in order to make the memory they currently occupy available for quick reuse. When control returns from the method call, the virtual machine has made its best effort to recycle all discarded objects.</p> <p>The name <code>gc</code> stands for "garbage collector". The virtual machine performs this recycling process automatically as needed, in a separate thread, even if the <code>gc</code> method is not invoked explicitly.</p> <p>The method <a href="#">System.gc()</a> is the conventional and convenient means of invoking this method.</p>	<p>public void <b>gc ()</b></p> <p>Since: API Level 1</p> <p>Indicates to the virtual machine that it would be a good time to run the garbage collector. Note that this is a hint only. There is no guarantee that the garbage collector will actually be run.</p>

Java™ 2 Platform Standard Edition 5.0 API Specification (java.lang.Runtime)	Android APIs (java.lang.Runtime)
<p><b>getLocalizedInputStream</b></p> <p><a href="#">@Deprecated</a>  public <a href="#">InputStream</a> <b>getLocalizedInputStream</b>(<a href="#">InputStream</a> in)</p> <p><b>Deprecated.</b> <i>As of JDK 1.1, the preferred way to translate a byte stream in the local encoding into a character stream in Unicode is via the <a href="#">InputStreamReader</a> and <a href="#">BufferedReader</a> classes.</i></p> <p>Creates a localized version of an input stream. This method takes an <a href="#">InputStream</a> and returns an <a href="#">InputStream</a> equivalent to the argument in all respects except that it is localized: as characters in the local character set are read from the stream, they are automatically converted from the local character set to Unicode.</p> <p>If the argument is already a localized stream, it may be returned as the result.</p> <p><b>Parameters:</b>  in - <a href="#">InputStream</a> to localize</p> <p><b>Returns:</b>  a localized input stream</p> <p><b>See Also:</b>  <a href="#">InputStream</a>, <a href="#">BufferedReader</a>, <a href="#">BufferedReader(java.io.Reader)</a>, <a href="#">InputStreamReader</a>, <a href="#">InputStreamReader(java.io.InputStream)</a></p>	<pre>public <a href="#">InputStream</a> <b>getLocalizedInputStream</b> (<a href="#">InputStream</a> stream)</pre> <p>Since: API Level 1</p> <p><b>This method is deprecated.</b>  Use <a href="#">InputStreamReader</a>.</p> <p>Returns the localized version of the specified input stream. The input stream that is returned automatically converts all characters from the local character set to Unicode after reading them from the underlying stream.</p> <p><b>Parameters</b></p> <p><i>stream</i>    the input stream to localize.</p> <p><b>Returns</b></p> <p>the localized input stream.</p>

Java™ 2 Platform Standard Edition 5.0 API Specification (java.lang.Runtime)	Android APIs (java.lang.Runtime)
<p><b>getLocalizedOutputStream</b></p> <p><a href="#">@Deprecated</a>  public <a href="#">OutputStream</a> <b>getLocalizedOutputStream</b>(<a href="#">OutputStream</a> out)</p> <p><b>Deprecated.</b> <i>As of JDK 1.1, the preferred way to translate a Unicode character stream into a byte stream in the local encoding is via the <a href="#">OutputStreamWriter</a>, <a href="#">BufferedWriter</a>, and <a href="#">PrintWriter</a> classes.</i> Creates a localized version of an output stream. This method takes an <a href="#">OutputStream</a> and returns an <a href="#">OutputStream</a> equivalent to the argument in all respects except that it is localized: as Unicode characters are written to the stream, they are automatically converted to the local character set.</p> <p>If the argument is already a localized stream, it may be returned as the result.</p> <p><b>Parameters:</b>  out - <a href="#">OutputStream</a> to localize</p> <p><b>Returns:</b>  a localized output stream</p> <p><b>See Also:</b>  <a href="#">OutputStream</a>, <a href="#">BufferedWriter</a>.<a href="#">BufferedWriter</a>(<a href="#">java.io.Writer</a>),  <a href="#">OutputStreamWriter</a>.<a href="#">OutputStreamWriter</a>(<a href="#">java.io.OutputStream</a>),  <a href="#">PrintWriter</a>.<a href="#">PrintWriter</a>(<a href="#">java.io.OutputStream</a>)</p>	<pre>public <a href="#">OutputStream</a> getLocalizedOutputStream (<a href="#">OutputStream</a> stream)</pre> <p>Since: API Level 1</p> <p><b>This method is deprecated.</b>  Use <a href="#">OutputStreamWriter</a>.</p> <p>Returns the localized version of the specified output stream. The output stream that is returned automatically converts all characters from Unicode to the local character set before writing them to the underlying stream.</p> <p><b>Parameters</b></p> <p><i>stream</i>    the output stream to localize.</p> <p><b>Returns</b></p> <p>the localized output stream.</p>

Java™ 2 Platform Standard Edition 5.0 API Specification (java.lang.Runtime)	Android APIs (java.lang.Runtime)
<p><b>getRuntime</b></p> <pre>public static <a href="#">Runtime</a> getRuntime()</pre> <p>Returns the runtime object associated with the current Java application. Most of the methods of class <code>Runtime</code> are instance methods and must be invoked with respect to the current runtime object.</p> <p><b>Returns:</b> the <code>Runtime</code> object associated with the current Java application.</p>	<pre>public static <a href="#">Runtime</a> getRuntime ()</pre> <p>Since: API Level 1</p> <p>Returns the single <code>Runtime</code> instance.</p> <p><b>Returns</b> the <code>Runtime</code> object for the current application.</p>

Java™ 2 Platform Standard Edition 5.0 API Specification (java.lang.Runtime)	Android APIs (java.lang.Runtime)
<p><b>halt</b></p> <pre>public void halt(int status)</pre> <p>Forcibly terminates the currently running Java virtual machine. This method never returns normally.</p> <p>This method should be used with extreme caution. Unlike the <a href="#">exit</a> method, this method does not cause shutdown hooks to be started and does not run uninvoked finalizers if finalization-on-exit has been enabled. If the shutdown sequence has already been initiated then this method does not wait for any running shutdown hooks or finalizers to finish their work.</p> <p><b>Parameters:</b>  status - Termination status. By convention, a nonzero status code indicates abnormal termination. If the <a href="#">exit</a> (equivalently, <a href="#">System.exit</a>) method has already been invoked then this status code will override the status code passed to that method.</p> <p><b>Throws:</b>  <a href="#">SecurityException</a> - If a security manager is present and its <a href="#">checkExit</a> method does not permit an exit with the specified status</p> <p><b>Since:</b>  1.3</p> <p><b>See Also:</b>  <a href="#">exit(int)</a>, <a href="#">addShutdownHook(java.lang.Thread)</a>,  <a href="#">removeShutdownHook(java.lang.Thread)</a></p>	<pre>public void halt(int code)</pre> <p>Since: API Level 1</p> <p>Causes the virtual machine to stop running, and the program to exit. Neither shutdown hooks nor finalizers are run before.</p> <p><b>Parameters</b></p> <p>code the return code. By convention, non-zero return codes indicate abnormal terminations.</p> <p><b>Throws</b></p> <p><a href="#">SecurityException</a> if the current <code>SecurityManager</code> does not allow the running thread to terminate the virtual machine.</p> <p><b>See Also</b></p> <p><a href="#">checkExit(int)</a>  <a href="#">addShutdownHook(Thread)</a>  <a href="#">removeShutdownHook(Thread)</a>  <a href="#">runFinalizersOnExit(boolean)</a></p>



Java™ 2 Platform Standard Edition 5.0 API Specification (java.lang.Runtime)	Android APIs (java.lang.Runtime)
<p><b>load</b></p> <pre>public void load(<a href="#">String</a> filename)</pre> <p>Loads the specified filename as a dynamic library. The filename argument must be a complete path name. From java_g it will automatically insert "_g" before the ".so" (for example <code>Runtime.getRuntime().load("/home/avh/lib/libX11.so");</code>).</p> <p>First, if there is a security manager, its <code>checkLink</code> method is called with the <code>filename</code> as its argument. This may result in a security exception.</p> <p>This is similar to the method <a href="#">loadLibrary(String)</a>, but it accepts a general file name as an argument rather than just a library name, allowing any file of native code to be loaded.</p> <p>The method <a href="#">System.load(String)</a> is the conventional and convenient means of invoking this method.</p> <p><b>Parameters:</b>  <code>filename</code> - the file to load.</p> <p><b>Throws:</b>  <a href="#">SecurityException</a> - if a security manager exists and its <code>checkLink</code> method doesn't allow loading of the specified dynamic library  <a href="#">UnsatisfiedLinkError</a> - if the file does not exist.  <a href="#">NullPointerException</a> - if <code>filename</code> is null</p> <p><b>See Also:</b>  <a href="#">getRuntime()</a>, <a href="#">SecurityException</a>,  <a href="#">SecurityManager.checkLink(java.lang.String)</a></p>	<pre>public void load (<a href="#">String</a> pathName)</pre> <p>Since: API Level 1</p> <p>Loads and links the dynamic library that is identified through the specified path. This method is similar to <a href="#">loadLibrary(String)</a>, but it accepts a full path specification whereas <code>loadLibrary</code> just accepts the name of the library to load.</p> <p><b>Parameters</b></p> <p><i>pathName</i>    the absolute (platform dependent) path to the library to load.</p> <p><b>Throws</b></p> <p><a href="#">UnsatisfiedLinkError</a>    if the library can not be loaded.</p> <p><a href="#">SecurityException</a>    if the current <code>SecurityManager</code> does not allow to load the library.</p> <p><b>See Also</b></p> <p><a href="#">checkLink(String)</a></p>

<b>Java™ 2 Platform Standard Edition 5.0 API Specification</b> <b>(java.lang.Runtime)</b>	<b>Android APIs</b> <b>(java.lang.Runtime)</b>
<p><b>loadLibrary</b></p> <pre>public void loadLibrary(<a href="#">String</a> libname)</pre> <p>Loads the dynamic library with the specified library name. A file containing native code is loaded from the local file system from a place where library files are conventionally obtained. The details of this process are implementation-dependent. The mapping from a library name to a specific filename is done in a system-specific manner.</p> <p>First, if there is a security manager, its <code>checkLink</code> method is called with the <code>libname</code> as its argument. This may result in a security exception.</p> <p>The method <a href="#">System.loadLibrary(String)</a> is the conventional and convenient means of invoking this method. If native methods are to be used in the implementation of a class, a standard strategy is to put the native code in a library file (call it <code>LibFile</code>) and then to put a static initializer:</p> <pre>static { System.loadLibrary("LibFile"); }</pre> <p>within the class declaration. When the class is loaded and initialized, the necessary native code implementation for the native methods will then be loaded as well.</p> <p>If this method is called more than once with the same library name, the second and subsequent calls are ignored.</p> <p><b>Parameters:</b>  <code>libname</code> - the name of the library.</p> <p><b>Throws:</b>  <a href="#">SecurityException</a> - if a security manager exists and its <code>checkLink</code> method doesn't allow loading of the specified dynamic library  <a href="#">UnsatisfiedLinkError</a> - if the library does not exist.</p>	<pre>public void loadLibrary(<a href="#">String</a> libName)</pre> <p>Since: API Level 1</p> <p>Loads and links the library with the specified name. The mapping of the specified library name to the full path for loading the library is implementation-dependent.</p> <p><b>Parameters</b></p> <p><i>libName</i>    the name of the library to load.</p> <p><b>Throws</b></p> <p><a href="#">UnsatisfiedLinkError</a>    if the library can not be loaded.</p> <p><a href="#">SecurityException</a>    if the current <code>SecurityManager</code> does not allow to load the library.</p> <p><b>See Also</b></p> <p><a href="#">checkLink(String)</a></p>

Java™ 2 Platform Standard Edition 5.0 API Specification (java.lang.Runtime)	Android APIs (java.lang.Runtime)
<a href="#">NullPointerException</a> - if libname is null <b>See Also:</b> <a href="#">SecurityException</a> , <a href="#">SecurityManager.checkLink(java.lang.String)</a>	
<b>maxMemory</b>  public long <b>maxMemory</b> () Returns the maximum amount of memory that the Java virtual machine will attempt to use. If there is no inherent limit then the value <a href="#">Long.MAX_VALUE</a> will be returned. <b>Returns:</b> the maximum amount of memory that the virtual machine will attempt to use, measured in bytes <b>Since:</b> 1.4	public long <b>maxMemory</b> () Since: API Level 1  Returns the maximum amount of memory that may be used by the virtual machine, or Long.MAX_VALUE if there is no such limit.  <b>Returns</b> the maximum amount of memory that the virtual machine will try to allocate, measured in bytes.

Java™ 2 Platform Standard Edition 5.0 API Specification ( <a href="#">java.lang.Runtime</a> )	Android APIs ( <a href="#">java.lang.Runtime</a> )
<p><b>removeShutdownHook</b></p> <p>public boolean <b>removeShutdownHook</b>(<a href="#">Thread</a> hook)</p> <p>De-registers a previously-registered virtual-machine shutdown hook.</p> <p><b>Parameters:</b> hook - the hook to remove</p> <p><b>Returns:</b> true if the specified hook had previously been registered and was successfully de-registered, false otherwise.</p> <p><b>Throws:</b> <a href="#">IllegalStateException</a> - If the virtual machine is already in the process of shutting down <a href="#">SecurityException</a> - If a security manager is present and it denies <a href="#">RuntimePermission</a>( "shutdownHooks" )</p> <p><b>Since:</b> 1.3</p> <p><b>See Also:</b> <a href="#">addShutdownHook( java.lang.Thread ), exit(int)</a></p>	<p>public boolean <b>removeShutdownHook</b> (<a href="#">Thread</a> hook)</p> <p>Since: API Level 1</p> <p>Unregisters a previously registered virtual machine shutdown hook.</p> <p><b>Parameters</b> hook the shutdown hook to remove.</p> <p><b>Returns</b> true if the hook has been removed successfully; false otherwise.</p> <p><b>Throws</b> <a href="#">IllegalStateException</a> if the virtual machine is already shutting down. <a href="#">SecurityException</a> if a SecurityManager is registered and the calling code doesn't have the <a href="#">RuntimePermission</a>("shutdownHooks").</p>

<b>Java™ 2 Platform Standard Edition 5.0 API Specification</b> <b>(java.lang.Runtime)</b>	<b>Android APIs</b> <b>(java.lang.Runtime)</b>
<p><b>runFinalization</b></p> <pre>public void runFinalization()</pre> <p>Runs the finalization methods of any objects pending finalization. Calling this method suggests that the Java virtual machine expend effort toward running the <code>finalize</code> methods of objects that have been found to be discarded but whose <code>finalize</code> methods have not yet been run. When control returns from the method call, the virtual machine has made a best effort to complete all outstanding finalizations.</p> <p>The virtual machine performs the finalization process automatically as needed, in a separate thread, if the <code>runFinalization</code> method is not invoked explicitly.</p> <p>The method <a href="#">System.runFinalization()</a> is the conventional and convenient means of invoking this method.</p> <p><b>See Also:</b>  <a href="#">Object.finalize()</a></p>	<pre>public void runFinalization ()</pre> <p>Since: API Level 1</p> <p>Provides a hint to the virtual machine that it would be useful to attempt to perform any outstanding object finalization.</p>

<p>Java™ 2 Platform Standard Edition 5.0 API Specification (java.lang.Runtime)</p>	<p>Android APIs (java.lang.Runtime)</p>
<p><b>runFinalizersOnExit</b></p> <p><a href="#">@Deprecated</a>  public static void <b>runFinalizersOnExit</b>(boolean value)</p> <p><b>Deprecated.</b> <i>This method is inherently unsafe. It may result in finalizers being called on live objects while other threads are concurrently manipulating those objects, resulting in erratic behavior or deadlock.</i></p> <p>Enable or disable finalization on exit; doing so specifies that the finalizers of all objects that have finalizers that have not yet been automatically invoked are to be run before the Java runtime exits. By default, finalization on exit is disabled.</p> <p>If there is a security manager, its <code>checkExit</code> method is first called with 0 as its argument to ensure the exit is allowed. This could result in a <code>SecurityException</code>.</p> <p><b>Parameters:</b>  value - true to enable finalization on exit, false to disable</p> <p><b>Throws:</b>  <a href="#">SecurityException</a> - if a security manager exists and its <code>checkExit</code> method doesn't allow the exit.</p> <p><b>Since:</b>  JDK1.1</p> <p><b>See Also:</b>  <a href="#">exit(int)</a>, <a href="#">gc()</a>, <a href="#">SecurityManager.checkExit(int)</a></p>	<p>public static void <b>runFinalizersOnExit</b> (boolean run)</p> <p>Since: API Level 1</p> <p><b>This method is deprecated.</b>  This method is unsafe.</p> <p>Sets the flag that indicates whether all objects are finalized when the virtual machine is about to exit. Note that all finalization which occurs when the system is exiting is performed after all running threads have been terminated.</p> <p><b>Parameters</b>  run true to enable finalization on exit, false to disable it.</p>

Java™ 2 Platform Standard Edition 5.0 API Specification ( <code>java.lang.Runtime</code> )	Android APIs ( <code>java.lang.Runtime</code> )
<p><b>totalMemory</b></p> <pre>public long totalMemory()</pre> <p>Returns the total amount of memory in the Java virtual machine. The value returned by this method may vary over time, depending on the host environment.</p> <p>Note that the amount of memory required to hold an object of any given type may be implementation-dependent.</p> <p><b>Returns:</b> the total amount of memory currently available for current and future objects, measured in bytes.</p>	<pre>public long totalMemory ()</pre> <p>Since: API Level 1</p> <p>Returns the total amount of memory which is available to the running program.</p> <p><b>Returns</b> the total amount of memory, measured in bytes.</p>
<p><b>traceInstructions</b></p> <pre>public void traceInstructions(boolean on)</pre> <p>Enables/Disables tracing of instructions. If the <code>boolean</code> argument is <code>true</code>, this method suggests that the Java virtual machine emit debugging information for each instruction in the virtual machine as it is executed. The format of this information, and the file or other output stream to which it is emitted, depends on the host environment. The virtual machine may ignore this request if it does not support this feature. The destination of the trace output is system dependent.</p> <p>If the <code>boolean</code> argument is <code>false</code>, this method causes the virtual machine to stop performing the detailed instruction trace it is performing.</p> <p><b>Parameters:</b> <code>on</code> - <code>true</code> to enable instruction tracing; <code>false</code> to disable this feature.</p>	<pre>public void traceInstructions (boolean enable)</pre> <p>Since: API Level 1</p> <p>Switches the output of debug information for instructions on or off. On Android, this method does nothing.</p> <p><b>Parameters</b> <code>enable</code> <code>true</code> to switch tracing on, <code>false</code> to switch it off.</p>

Java™ 2 Platform Standard Edition 5.0 API Specification ( <code>java.lang.Runtime</code> )	Android APIs ( <code>java.lang.Runtime</code> )
<p><b>traceMethodCalls</b></p> <pre>public void <b>traceMethodCalls</b>(boolean on)</pre> <p>Enables/Disables tracing of method calls. If the <code>boolean</code> argument is <code>true</code>, this method suggests that the Java virtual machine emit debugging information for each method in the virtual machine as it is called. The format of this information, and the file or other output stream to which it is emitted, depends on the host environment. The virtual machine may ignore this request if it does not support this feature.</p> <p>Calling this method with argument <code>false</code> suggests that the virtual machine cease emitting per-call debugging information.</p> <p><b>Parameters:</b></p> <p><code>on</code> - <code>true</code> to enable instruction tracing; <code>false</code> to disable this feature.</p>	<pre>public void <b>traceMethodCalls</b> (boolean enable)</pre> <p>Since: API Level 1</p> <p>Switches the output of debug information for methods on or off.</p> <p><b>Parameters</b></p> <p><code>enable</code>    <code>true</code> to switch tracing on, <code>false</code> to switch it off.</p>



## Exhibit Copyright-G

Java™ 2 Platform Standard Edition 5.0 API Specification ( <a href="#">java.security.ProtectionDomain</a> )	Android Source Code <sup>3</sup> <a href="#">dalvik/libcore/security/src/main/java/java/security/ProtectionDomain.java</a>
<div data-bbox="197 310 663 358" data-label="Section-Header"> <h3>Constructor Summary</h3> </div> <div data-bbox="197 375 1003 516" data-label="Text"> <p><a href="#">ProtectionDomain</a>(<a href="#">CodeSource</a> codesource, <a href="#">PermissionCollection</a> permissions) Creates a new ProtectionDomain with the given CodeSource and Permissions.</p> </div>	<pre>public ProtectionDomain(CodeSource cs, PermissionCollection permissions) {     this.codeSource = cs;     if (permissions != null) {         permissions.setReadOnly();     }     this.permissions = permissions;     //this.classLoader = null;     //this.principals = null;     //dynamicPerms = false; }</pre>
<div data-bbox="197 659 1003 821" data-label="Text"> <p><a href="#">ProtectionDomain</a>(<a href="#">CodeSource</a> codesource, <a href="#">PermissionCollection</a> permissions, <a href="#">ClassLoader</a> classloader, <a href="#">Principal</a>[] principals) Creates a new ProtectionDomain qualified by the given CodeSource, Permissions, ClassLoader and array of Principals.</p> </div>	<pre>public ProtectionDomain(CodeSource cs, PermissionCollection permissions, ClassLoader cl, Principal[] principals) {     this.codeSource = cs;     if (permissions != null) {         permissions.setReadOnly();     }     this.permissions = permissions;     this.classLoader = cl;     if (principals != null) {         this.principals = new Principal[principals.length];         System.arraycopy(principals, 0, this.principals, 0, this.principals.length);     }     dynamicPerms = true; }</pre>
<div data-bbox="197 1305 571 1354" data-label="Section-Header"> <h3>Method Summary</h3> </div> <div data-bbox="197 1370 1003 1468" data-label="Text"> <p><a href="#">ClassLoader</a> <a href="#">getClassLoader</a>() Returns the ClassLoader of this domain.</p> </div>	<pre>public final ClassLoader getClassLoader() {     return classLoader; }</pre>

<a href="#">CodeSource</a>	<a href="#">getCodeSource()</a> Returns the CodeSource of this domain.	<pre>public final CodeSource getCodeSource() {     return codeSource; }</pre>
<a href="#">PermissionCollection</a>	<a href="#">getPermissions()</a> Returns the static permissions granted to this domain.	<pre>public final PermissionCollection getPermissions() {     return permissions; }</pre>
<a href="#">Principal[]</a>	<a href="#">getPrincipals()</a> Returns an array of principals for this domain.	<pre>public final Principal[] getPrincipals() {     if( principals == null ) {         return new Principal[0];     }     Principal[] tmp = new Principal[principals.length];     System.arraycopy(principals, 0, tmp, 0, tmp.length);     return tmp; }</pre>
boolean	<a href="#">implies(Permission permission)</a> Check and see if this ProtectionDomain implies the permissions expressed in the Permission object.	<pre>public boolean implies(Permission permission) {     // First, test with the Policy, as the     default Policy.implies()     // checks for both dynamic and static     collections of the     // ProtectionDomain passed...     if (dynamicPerms         &amp;&amp;         Policy.getAccessiblePolicy().implies(this,         permission)) {         return true;     } }</pre>
<a href="#">String</a>	<a href="#">toString()</a> Convert a ProtectionDomain to a String.	<pre>public String toString() {     StringBuilder buf = new StringBuilder(200);     buf.append("ProtectionDomain\n"); //\$NON- NLS-1\$     buf.append("CodeSource=").append( //\$NON- NLS-1\$         codeSource == null ? "&lt;null&gt;" : codeSource.toString()).append( //\$NON-NLS-1\$</pre>

```

        "\n"); //$NON-NLS-1$
        buf.append("ClassLoader=").append( //$NON-NLS-1$
            classLoader == null ? "<null>" :
            classLoader.toString()) //$NON-NLS-1$
            .append("\n"); //$NON-NLS-1$
        if (principals == null ||
            principals.length == 0) {
            buf.append("<no principals>\n");
        //$NON-NLS-1$
        } else {
            buf.append("Principals: <\n"); //$NON-NLS-1$
            for (int i = 0; i < principals.length;
                i++) {
                buf.append("\t").append( //$NON-NLS-1$
                    principals[i] == null ?
                    "<null>" : principals[i] //$NON-NLS-1$
                    .toString()).append("\n"); //$NON-NLS-1$
            }
            buf.append(">"); //$NON-NLS-1$
        }

        //permissions here
        buf.append("Permissions:\n"); //$NON-NLS-1$
        if (permissions == null) {
            buf.append("\t\t<no static
permissions>\n"); //$NON-NLS-1$
        } else {
            buf.append("\t\tstatic:
").append(permissions.toString()).append( //$NON-NLS-1$
                "\n"); //$NON-NLS-1$
        }

        if (dynamicPerms) {
            if (Policy.isSet()) {

```

	<pre>        PermissionCollection perms;         perms = Policy.getAccessiblePolicy().getPermissions(this);         if (perms == null) {             buf.append("\t\t&lt;no dynamic permissions&gt;\n"); //\$NON-NLS-1\$         } else {             buf.append("\t\tdynamic: ").append(perms.toString()) //\$NON-NLS-1\$                 .append("\n"); //\$NON- NLS-1\$         }         } else {             buf.append("\t\t&lt;no dynamic permissions&gt;\n"); //\$NON-NLS-1\$         }     }     return buf.toString(); }</pre>
--	--

## Exhibit Copyright-H

### ProtectionDomain.java from Android 2.2 ("Froyo")

```

/*
 * Licensed to the Apache Software Foundation (ASF) under one or more
 * contributor license agreements. See the NOTICE file distributed with
 * this work for additional information regarding copyright ownership.
 * The ASF licenses this file to You under the Apache License, Version 2.0
 * (the "License"); you may not use this file except in compliance with
 * the License. You may obtain a copy of the License at
 *
 *      http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */

package java.security;

/**
 * {@code ProtectionDomain} represents all permissions that are granted to a
 * specific code source. The {@link ClassLoader} associates each class with the
 * corresponding {@code ProtectionDomain}, depending on the location and the
 * certificates (encapsulates in {@link CodeSource}) it loads the code from.
 * <p>
 * A class belongs to exactly one protection domain and the protection domain
 * can not be changed during the lifetime of the class.
 */
public class ProtectionDomain {

    // CodeSource for this ProtectionDomain
    private CodeSource codeSource;

    // Static permissions for this ProtectionDomain
    private PermissionCollection permissions;

    // ClassLoader
    private ClassLoader classLoader;

    // Set of principals associated with this ProtectionDomain
    private Principal[] principals;

    // false if this ProtectionDomain was constructed with static
    // permissions, true otherwise.
    private boolean dynamicPerms;

```

```

/**
 * Constructs a new instance of {@code ProtectionDomain} with the specified
 * code source and the specified static permissions.
 * <p>
 * If {@code permissions} is not {@code null}, the {@code permissions}
 * collection is made immutable by calling
 * {@link PermissionCollection#setReadOnly()} and it is considered as
 * granted statically to this {@code ProtectionDomain}.
 * <p>
 * The policy will not be consulted by access checks against this {@code
 * ProtectionDomain}.
 * <p>
 * If {@code permissions} is {@code null}, the method {@link
 * ProtectionDomain#implies(Permission)} always returns {@code false}.
 *
 * @param cs
 *         the code source associated with this domain, maybe {@code
 *         null}.
 * @param permissions
 *         the {@code PermissionCollection} containing all permissions to
 *         be statically granted to this {@code ProtectionDomain}, maybe
 *         {@code null}.
 */
public ProtectionDomain(CodeSource cs, PermissionCollection permissions) {
    this.codeSource = cs;
    if (permissions != null) {
        permissions.setReadOnly();
    }
    this.permissions = permissions;
    //this.classLoader = null;
    //this.principals = null;
    //dynamicPerms = false;
}

/**
 * Constructs a new instance of {@code ProtectionDomain} with the specified
 * code source, the permissions, the class loader and the principals.
 * <p>
 * If {@code permissions} is {@code null}, and access checks are performed
 * against this protection domain, the permissions defined by the policy are
 * consulted. If {@code permissions} is not {@code null}, the {@code
 * permissions} collection is made immutable by calling
 * {@link PermissionCollection#setReadOnly()}. If access checks are
 * performed, the policy and the provided permission collection are checked.
 * <p>
 * External modifications of the provided {@code principals} array has no
 * impact on this {@code ProtectionDomain}.
 *

```

```

* @param cs
*     the code source associated with this domain, maybe {@code
*     null}.
* @param permissions
*     the permissions associated with this domain, maybe {@code
*     null}.
* @param cl
*     the class loader associated with this domain, maybe {@code
*     null}.
* @param principals
*     the principals associated with this domain, maybe {@code
*     null}.
*/
public ProtectionDomain(CodeSource cs, PermissionCollection permissions,
    ClassLoader cl, Principal[] principals) {
    this.codeSource = cs;
    if (permissions != null) {
        permissions.setReadOnly();
    }
    this.permissions = permissions;
    this.classLoader = cl;
    if (principals != null) {
        this.principals = new Principal[principals.length];
        System.arraycopy(principals, 0, this.principals, 0,
            this.principals.length);
    }
    dynamicPerms = true;
}

/**
 * Returns the {@code ClassLoader} associated with this {@code
 * ProtectionDomain}.
 *
 * @return the {@code ClassLoader} associated with this {@code
 *     ProtectionDomain}, maybe {@code null}.
 */
public final ClassLoader getClassLoader() {
    return classLoader;
}

/**
 * Returns the {@code CodeSource} of this {@code ProtectionDomain}.
 *
 * @return the {@code CodeSource} of this {@code ProtectionDomain}, maybe
 *     {@code null}.
 */
public final CodeSource getCodeSource() {
    return codeSource;
}

```

```

/**
 * Returns the static permissions that are granted to this {@code
 * ProtectionDomain}.
 *
 * @return the static permissions that are granted to this {@code
 *         ProtectionDomain}, maybe {@code null}.
 */
public final PermissionCollection getPermissions() {
    return permissions;
}

/**
 * Returns the principals associated with this {@code ProtectionDomain}.
 * Modifications of the returned {@code Principal} array has no impact on
 * this {@code ProtectionDomain}.
 *
 * @return the principals associated with this {@code ProtectionDomain}.
 */
public final Principal[] getPrincipals() {
    if( principals == null ) {
        return new Principal[0];
    }
    Principal[] tmp = new Principal[principals.length];
    System.arraycopy(principals, 0, tmp, 0, tmp.length);
    return tmp;
}

/**
 * Indicates whether the specified permission is implied by this {@code
 * ProtectionDomain}.
 *
 * <p>
 * If this {@code ProtectionDomain} was constructed with
 * {@link #ProtectionDomain(CodeSource, PermissionCollection)}, the
 * specified permission is only checked against the permission collection
 * provided in the constructor. If {@code null} was provided, {@code false}
 * is returned.
 *
 * <p>
 * If this {@code ProtectionDomain} was constructed with
 * {@link #ProtectionDomain(CodeSource, PermissionCollection, ClassLoader, Principal[])}
 * , the specified permission is checked against the policy and the
 * permission collection provided in the constructor.
 *
 * @param permission
 *         the permission to check against the domain.
 * @return {@code true} if the specified {@code permission} is implied by
 *         this {@code ProtectionDomain}, {@code false} otherwise.
 */
public boolean implies(Permission permission) {

```



```

// First, test with the Policy, as the default Policy.implies()
// checks for both dynamic and static collections of the
// ProtectionDomain passed...
if (dynamicPerms
    && Policy.getAccessiblePolicy().implies(this, permission)) {
    return true;
}

// ... and we get here if
// either the permissions are static
// or Policy.implies() did not check for static permissions
// or the permission is not implied
return permissions == null ? false : permissions.implies(permission);
}

/**
 * Returns a string containing a concise, human-readable description of the
 * this {@code ProtectionDomain}.
 *
 * @return a printable representation for this {@code ProtectionDomain}.
 */
@Override
public String toString() {
    StringBuilder buf = new StringBuilder(200);
    buf.append("ProtectionDomain\n"); //$NON-NLS-1$
    buf.append("CodeSource=").append( //$NON-NLS-1$
        codeSource == null ? "<null>" : codeSource.toString()).append( //$NON-NLS-1$
        "\n"); //$NON-NLS-1$
    buf.append("ClassLoader=").append( //$NON-NLS-1$
        classLoader == null ? "<null>" : classLoader.toString()) //$NON-NLS-1$
        .append("\n"); //$NON-NLS-1$
    if (principals == null || principals.length == 0) {
        buf.append("<no principals>\n"); //$NON-NLS-1$
    } else {
        buf.append("Principals: <\n"); //$NON-NLS-1$
        for (int i = 0; i < principals.length; i++) {
            buf.append("\t").append( //$NON-NLS-1$
                principals[i] == null ? "<null>" : principals[i] //$NON-NLS-1$
                    .toString()).append("\n"); //$NON-NLS-1$
        }
        buf.append(">"); //$NON-NLS-1$
    }

    //permissions here
    buf.append("Permissions:\n"); //$NON-NLS-1$
    if (permissions == null) {
        buf.append("\t\t<no static permissions>\n"); //$NON-NLS-1$
    } else {
        buf.append("\t\tstatic: ").append(permissions.toString()).append( //$NON-NLS-1$

```

```

        "\n"); //$NON-NLS-1$
    }
    if (dynamicPerms) {
        if (Policy.isSet()) {
            PermissionCollection perms;
            perms = Policy.getAccessiblePolicy().getPermissions(this);
            if (perms == null) {
                buf.append("\t\t<no dynamic permissions>\n"); //$NON-NLS-1$
            } else {
                buf.append("\t\tdynamic: ").append(perms.toString()) //$NON-NLS-1$
                    .append("\n"); //$NON-NLS-1$
            }
        } else {
            buf.append("\t\t<no dynamic permissions>\n"); //$NON-NLS-1$
        }
    }
    return buf.toString();
}
}

```

## **Exhibit Copyright-I**

readme.txt from SGH-I897\_OpenSource.tar.gz

How to build

1. Get android open source.  
: version info - Android eclair 2.1 (android-2.1\_r2)  
( Download site : <http://source.android.com> )
2. Overwrite modules that you want to build.
3. Add the following lines at the end of build/target/board/generic/BoardConfig.mk  
  
BOARD\_HAVE\_BLUETOOTH := true  
BT\_USE\_BTL\_IF := true  
BT\_ALT\_STACK := true  
BRCM\_BTL\_INCLUDE\_A2DP := true  
BRCM\_BT\_USE\_BTL\_IF := true
4. make update-api
5. make

## Exhibit Copyright-J

PolicyNodeImpl.jad (decompiled version of Oracle PolicyNodeImpl.class) [spacing adjusted for comparison]	PolicyNodeImpl.java (Android version) [spacing adjusted for comparison]
<pre>// Decompiled by Jad v1.5.8g. Copyright 2001 Pavel Kouznetsov. // Jad home page: http://www.kpdus.com/jad.html // Decompiler options: fieldsfirst nonlb // Source File Name:    PolicyNodeImpl.java  package sun.security.provider.certpath;  import java.security.cert.PolicyNode; import java.util.Collections; import java.util.HashSet; import java.util.Iterator; import java.util.Set;  final class PolicyNodeImpl     implements PolicyNode {      private static final String ANY_POLICY = "2.5.29.32.0";     private PolicyNodeImpl mParent;     private HashSet mChildren;     private String mValidPolicy;     private HashSet mQualifierSet;     private boolean mCriticalityIndicator;     private HashSet mExpectedPolicySet;     private boolean mOriginalExpectedPolicySet;     private int mDepth;     private boolean isImmutable;      PolicyNodeImpl(PolicyNodeImpl policynodeimpl, String s, Set set, boolean flag, Set set1, boolean flag1) {         isImmutable = false;         mParent = policynodeimpl;         mChildren = new HashSet();         if(s != null)             mValidPolicy = s;         else             mValidPolicy = "";          if(set != null)             mQualifierSet = new HashSet(set);         else             mQualifierSet = new HashSet();          mCriticalityIndicator = flag; </pre>	<pre>/*  * Licensed to the Apache Software Foundation (ASF) under one or more  * contributor license agreements. See the NOTICE file distributed with  * this work for additional information regarding copyright ownership.  * The ASF licenses this file to You under the Apache License, Version 2.0  * (the "License"); you may not use this file except in compliance with  * the License. You may obtain a copy of the License at  *  *     http://www.apache.org/licenses/LICENSE-2.0  *  * Unless required by applicable law or agreed to in writing, software  * distributed under the License is distributed on an "AS IS" BASIS,  * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.  * See the License for the specific language governing permissions and  * limitations under the License.  */  package org.apache.harmony.security.tests.support.cert;  import java.security.cert.PolicyNode; import java.util.*;  public class PolicyNodeImpl implements PolicyNode {      private static final String ANY_POLICY = "2.5.29.32.0";     private PolicyNodeImpl mParent;     private HashSet mChildren;     private String mValidPolicy;     private HashSet mQualifierSet;     private boolean mCriticalityIndicator;     private HashSet mExpectedPolicySet;     private boolean mOriginalExpectedPolicySet;     private int mDepth;     private boolean isImmutable;      public PolicyNodeImpl(PolicyNodeImpl policynodeimpl, String s, Set set, boolean flag, Set set1, boolean flag1) {         isImmutable = false;         mParent = policynodeimpl;         mChildren = new HashSet();         if(s != null) {             mValidPolicy = s;         } else {             mValidPolicy = "";         }         if(set != null) {             mQualifierSet = new HashSet(set);         } else {             mQualifierSet = new HashSet();         }         mCriticalityIndicator = flag; </pre>

PolicyNodeImpl.jad (decompiled version of Oracle PolicyNodeImpl.class) [spacing adjusted for comparison]	PolicyNodeImpl.java (Android version) [spacing adjusted for comparison]
<pre>         if(set1 != null)             mExpectedPolicySet = new HashSet(set1);         else             mExpectedPolicySet = new HashSet();         mOriginalExpectedPolicySet = !flag1;         if(mParent != null) {             mDepth = mParent.getDepth() + 1;             mParent.addChild(this);         } else {             mDepth = 0;         }     }      PolicyNodeImpl(PolicyNodeImpl polycynodeimpl, PolicyNodeImpl     polycynodeimpl1) {         this(polycynodeimpl, polycynodeimpl1.mValidPolicy, ((Set)         (polycynodeimpl1.mQualifierSet)), polycynodeimpl1.mCriticalityIndicator,         ((Set) (polycynodeimpl1.mExpectedPolicySet)), false);     }      public PolicyNode getParent() {         return mParent;     }      public Iterator getChildren() {         return Collections.unmodifiableSet(mChildren).iterator();     }      public int getDepth() {         return mDepth;     }      public String getValidPolicy() {         return mValidPolicy;     }      public Set getPolicyQualifiers() {         return Collections.unmodifiableSet(mQualifierSet);     }      public Set getExpectedPolicies() {         return Collections.unmodifiableSet(mExpectedPolicySet);     }      public boolean isCritical() {         return mCriticalityIndicator;     }      public String toString() {         StringBuffer stringbuffer = new StringBuffer(asString());         for(Iterator iterator = getChildren(); iterator.hasNext();         stringbuffer.append((PolicyNodeImpl)iterator.next()));         return stringbuffer.toString();     }      boolean isImmutable() {         return isImmutable;     } </pre>	<pre>         if(set1 != null) {             mExpectedPolicySet = new HashSet(set1);         } else {             mExpectedPolicySet = new HashSet();         }         mOriginalExpectedPolicySet = !flag1;         if(mParent != null) {             mDepth = mParent.getDepth() + 1;             mParent.addChild(this);         } else {             mDepth = 0;         }     }      PolicyNodeImpl(PolicyNodeImpl polycynodeimpl,     PolicyNodeImpl polycynodeimpl1) {         this(polycynodeimpl, polycynodeimpl1.mValidPolicy, ((Set)         (polycynodeimpl1.mQualifierSet)), polycynodeimpl1.mCriticalityIndicator,         ((Set) (polycynodeimpl1.mExpectedPolicySet)), false);     }      public PolicyNode getParent() {         return mParent;     }      public Iterator getChildren() {         return Collections.unmodifiableSet(mChildren).iterator();     }      public int getDepth() {         return mDepth;     }      public String getValidPolicy() {         return mValidPolicy;     }      public Set getPolicyQualifiers() {         return Collections.unmodifiableSet(mQualifierSet);     }      public Set getExpectedPolicies() {         return Collections.unmodifiableSet(mExpectedPolicySet);     }      public boolean isCritical() {         return mCriticalityIndicator;     }      public String toString() {         StringBuffer stringbuffer = new StringBuffer(asString());         for(Iterator iterator = getChildren(); iterator.hasNext();         stringbuffer.append((PolicyNodeImpl)iterator.next()));         return stringbuffer.toString();     }      boolean isImmutable() {         return isImmutable;     } </pre>

PolicyNodeImpl.jad (decompiled version of Oracle PolicyNodeImpl.class) [spacing adjusted for comparison]	PolicyNodeImpl.java (Android version) [spacing adjusted for comparison]
<pre> void setImmutable() {     if(!immutable)         return;     PolicyNodeImpl polycnodeimpl;     for(Iterator iterator = mChildren.iterator(); iterator.hasNext();     polycnodeimpl.setImmutable())         polycnodeimpl = (PolicyNodeImpl)iterator.next();      immutable = true; }  private void addChild(PolicyNodeImpl polycnodeimpl) {     if(!immutable) {         throw new IllegalStateException("PolicyNode is immutable");     } else {         mChildren.add(polycnodeimpl);         return;     } }  void addExpectedPolicy(String s) {     if(!immutable)         throw new IllegalStateException("PolicyNode is immutable");     if(mOriginalExpectedPolicySet) {         mExpectedPolicySet.clear();         mOriginalExpectedPolicySet = false;     }     mExpectedPolicySet.add(s); }  void prune(int i) {     if(!immutable)         throw new IllegalStateException("PolicyNode is immutable");     if(mChildren.size() == 0)         return;     Iterator iterator = mChildren.iterator();     do {         if(!iterator.hasNext())             break;         PolicyNodeImpl polycnodeimpl = (PolicyNodeImpl)iterator.next();         polycnodeimpl.prune(i);         if(polycnodeimpl.mChildren.size() == 0 &amp;&amp; i &gt; mDepth + 1)             iterator.remove();     } while(true); }  void deleteChild(PolicyNode polycnode) {     if(!immutable) {         throw new IllegalStateException("PolicyNode is immutable");     } else {         mChildren.remove(polycnode);         return;     } }  PolicyNodeImpl copyTree() {     return copyTree(null); } </pre>	<pre> void setImmutable() {     if(!immutable) return;      PolicyNodeImpl polycnodeimpl;     for(Iterator iterator = mChildren.iterator(); iterator.hasNext();     polycnodeimpl.setImmutable())         polycnodeimpl = (PolicyNodeImpl)iterator.next();      immutable = true; }  private void addChild(PolicyNodeImpl polycnodeimpl) {     if(!immutable) {         throw new IllegalStateException("PolicyNode is immutable");     } else {         mChildren.add(polycnodeimpl);         return;     } }  void addExpectedPolicy(String s) {     if(!immutable)         throw new IllegalStateException("PolicyNode is immutable");     if(mOriginalExpectedPolicySet) {         mExpectedPolicySet.clear();         mOriginalExpectedPolicySet = false;     }     mExpectedPolicySet.add(s); }  void prune(int i) {     if(!immutable)         throw new IllegalStateException("PolicyNode is immutable");     if(mChildren.size() == 0)         return;     Iterator iterator = mChildren.iterator();     do {         if(!iterator.hasNext()) break;         PolicyNodeImpl polycnodeimpl = (PolicyNodeImpl)iterator.next();         polycnodeimpl.prune(i);         if(polycnodeimpl.mChildren.size() == 0 &amp;&amp; i &gt; mDepth + 1)             iterator.remove();     } while(true); }  void deleteChild(PolicyNode polycnode) {     if(!immutable) {         throw new IllegalStateException("PolicyNode is immutable");     } else {         mChildren.remove(polycnode);         return;     } }  PolicyNodeImpl copyTree() {     return copyTree(null); } </pre>

PolicyNodeImpl.jad (decompiled version of Oracle PolicyNodeImpl.class) [spacing adjusted for comparison]	PolicyNodeImpl.java (Android version) [spacing adjusted for comparison]
<pre> private PolicyNodeImpl copyTree(PolicyNodeImpl polycynodeimpl) {     PolicyNodeImpl polycynodeimpl1 = new PolicyNodeImpl(polycynodeimpl, this);     PolicyNodeImpl polycynodeimpl2;     for(Iterator iterator = mChildren.iterator(); iterator.hasNext(); polycynodeimpl2.copyTree(polycynodeimpl1))         polycynodeimpl2 = (PolicyNodeImpl)iterator.next();      return polycynodeimpl1; }  Set getPolicyNodes(int i) {     HashSet hashset = new HashSet();     getPolicyNodes(i, ((Set) (hashset)));     return hashset; }  private void getPolicyNodes(int i, Set set) {     if(mDepth == i) {         set.add(this);     } else {         PolicyNodeImpl polycynodeimpl;         for(Iterator iterator = mChildren.iterator(); iterator.hasNext(); polycynodeimpl.getPolicyNodes(i, set))             polycynodeimpl = (PolicyNodeImpl)iterator.next();     } }  Set getPolicyNodesExpected(int i, String s, boolean flag) {     if(s.equals("2.5.29.32.0"))         return getPolicyNodes(i);     else         return getPolicyNodesExpectedHelper(i, s, flag); }  private Set getPolicyNodesExpectedHelper(int i, String s, boolean flag) {     HashSet hashset = new HashSet();     if(mDepth &lt; i) {         PolicyNodeImpl polycynodeimpl;         for(Iterator iterator = mChildren.iterator(); iterator.hasNext(); hashset.addAll(polycynodeimpl.getPolicyNodesExpectedHelper(i, s, flag)))             polycynodeimpl = (PolicyNodeImpl)iterator.next();      } else         if(flag) {             if(mExpectedPolicySet.contains("2.5.29.32.0"))                 hashset.add(this);         } else             if(mExpectedPolicySet.contains(s))                 hashset.add(this);     return hashset; }  Set getPolicyNodesValid(int i, String s) {     HashSet hashset = new HashSet(); </pre>	<pre> private PolicyNodeImpl copyTree(PolicyNodeImpl polycynodeimpl) {     PolicyNodeImpl polycynodeimpl1 = new PolicyNodeImpl(polycynodeimpl, this);     PolicyNodeImpl polycynodeimpl2;     for(Iterator iterator = mChildren.iterator(); iterator.hasNext(); polycynodeimpl2.copyTree(polycynodeimpl1))         polycynodeimpl2 = (PolicyNodeImpl)iterator.next();      return polycynodeimpl1; }  Set getPolicyNodes(int i) {     HashSet hashset = new HashSet();     getPolicyNodes(i, ((Set) (hashset)));     return hashset; }  private void getPolicyNodes(int i, Set set) {     if(mDepth == i) {         set.add(this);     } else {         PolicyNodeImpl polycynodeimpl;         for(Iterator iterator = mChildren.iterator(); iterator.hasNext(); polycynodeimpl.getPolicyNodes(i, set))             polycynodeimpl = (PolicyNodeImpl)iterator.next();     } }  Set getPolicyNodesExpected(int i, String s, boolean flag) {     if(s.equals("2.5.29.32.0"))         return getPolicyNodes(i);     else         return getPolicyNodesExpectedHelper(i, s, flag); }  private Set getPolicyNodesExpectedHelper(int i, String s, boolean flag) {     HashSet hashset = new HashSet();     if(mDepth &lt; i) {         PolicyNodeImpl polycynodeimpl;         for(Iterator iterator = mChildren.iterator(); iterator.hasNext(); hashset.addAll(polycynodeimpl.getPolicyNodesExpectedHelper(i, s, flag)))             polycynodeimpl = (PolicyNodeImpl)iterator.next();      } else if(flag) {         if(mExpectedPolicySet.contains("2.5.29.32.0"))             hashset.add(this);     } else if(mExpectedPolicySet.contains(s)) {         hashset.add(this);     }     return hashset; }  Set getPolicyNodesValid(int i, String s) {     HashSet hashset = new HashSet(); </pre>

PolicyNodeImpl.jad (decompiled version of Oracle PolicyNodeImpl.class) [spacing adjusted for comparison]	PolicyNodeImpl.java (Android version) [spacing adjusted for comparison]
<pre> if(mDepth &lt; i) {     PolicyNodeImpl polycnodeimpl;     for(Iterator iterator = mChildren.iterator(); iterator.hasNext(); hashset.addAll(polycnodeimpl.getPolicyNodesValid(i, s)))         polycnodeimpl = (PolicyNodeImpl)iterator.next();  } else if(mValidPolicy.equals(s))     hashset.add(this); return hashset; }  private static String policyToString(String s) {     if(s.equals("2.5.29.32.0"))         return "anyPolicy";     else         return s; }  String asString() {     if(mParent == null)         return "anyPolicy ROOT\n";     StringBuffer stringbuffer = new StringBuffer();     int i = 0;     for(int j = getDepth(); i &lt; j; i++)         stringbuffer.append(" ");      stringbuffer.append(policyToString(getValidPolicy()));     stringbuffer.append(" CRIT: ");     stringbuffer.append(isCritical());     stringbuffer.append(" EP: ");     for(Iterator iterator = getExpectedPolicies().iterator(); iterator.hasNext(); stringbuffer.append(" ")) {         String s = (String)iterator.next();         stringbuffer.append(policyToString(s));     }      stringbuffer.append(" (");     stringbuffer.append(getDepth());     stringbuffer.append(")\n");     return stringbuffer.toString(); } } </pre>	<pre> if(mDepth &lt; i) {     PolicyNodeImpl polycnodeimpl;     for(Iterator iterator = mChildren.iterator(); iterator.hasNext(); hashset.addAll(polycnodeimpl.getPolicyNodesValid(i, s)))         polycnodeimpl = (PolicyNodeImpl)iterator.next();  } else if(mValidPolicy.equals(s)) {     hashset.add(this); } return hashset; }  private static String policyToString(String s) {     if(s.equals("2.5.29.32.0")) {         return "anyPolicy";     } else {         return s;     } }  String asString() {     if(mParent == null)         return "anyPolicy ROOT\n";     StringBuffer stringbuffer = new StringBuffer();     int i = 0;     for(int j = getDepth(); i &lt; j; i++)         stringbuffer.append(" ");      stringbuffer.append(policyToString(getValidPolicy()));     stringbuffer.append(" CRIT: ");     stringbuffer.append(isCritical());     stringbuffer.append(" EP: ");     for(Iterator iterator = getExpectedPolicies().iterator(); iterator.hasNext(); stringbuffer.append(" ")) {         String s = (String)iterator.next();         stringbuffer.append(policyToString(s));     }      stringbuffer.append(" (");     stringbuffer.append(getDepth());     stringbuffer.append(")\n");     return stringbuffer.toString(); } } </pre>



**Exhibit Copyright-K**

AclEntryImpl.jad (decompiled version of Oracle AclEntryImpl.class) [spacing adjusted for comparison]	AclEntryImpl.java (Android version) [spacing adjusted for comparison]
<pre>// Decompiled by Jad v1.5.8g. Copyright 2001 Pavel Kouznetsov. // Jad home page: http://www.kpdus.com/jad.html // Decompiler options: fieldsfirst nonlb // Source File Name:   AclEntryImpl.java  package sun.security.acl;  import java.security.Principal; import java.security.acl.AclEntry; import java.security.acl.Group; import java.security.acl.Permission; import java.util.Enumeration; import java.util.Vector;  public class AclEntryImpl     implements AclEntry {      private Principal user;     private Vector permissionSet;     private boolean negative;      public AclEntryImpl(Principal principal) {         user = null;         permissionSet = new Vector(10, 10);         negative = false;         user = principal;     }      public AclEntryImpl() {         user = null;         permissionSet = new Vector(10, 10);         negative = false;     }      public boolean setPrincipal(Principal principal) {         if(user != null) {             return false;         } else {             user = principal;             return true;         }     }      public void setNegativePermissions() {</pre>	<pre>/*  * Licensed to the Apache Software Foundation (ASF) under one or more  * contributor license agreements. See the NOTICE file distributed with  * this work for additional information regarding copyright ownership.  * The ASF licenses this file to You under the Apache License, Version 2.0  * (the "License"); you may not use this file except in compliance with  * the License. You may obtain a copy of the License at  *  *     http://www.apache.org/licenses/LICENSE-2.0  *  * Unless required by applicable law or agreed to in writing, software  * distributed under the License is distributed on an "AS IS" BASIS,  * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.  * See the License for the specific language governing permissions and  * limitations under the License.  */  package org.apache.harmony.security.tests.support.acl;  import java.security.Principal; import java.security.acl.*; import java.util.Enumeration; import java.util.Vector;  /**  * Additional class for verification AclEntry interface  */ public class AclEntryImpl implements AclEntry {      private Principal user;     private Vector permissionSet;     private boolean negative;      public AclEntryImpl(Principal principal) {         user = null;         permissionSet = new Vector(10, 10);         negative = false;         user = principal;     }      public AclEntryImpl() {         user = null;         permissionSet = new Vector(10, 10);         negative = false;     }      public boolean setPrincipal(Principal principal) {         if(user != null) {             return false;         } else {             user = principal;             return true;         }     }      public void setNegativePermissions() {</pre>

AclEntryImpl.jad (decompiled version of Oracle AclEntryImpl.class) [spacing adjusted for comparison]	AclEntryImpl.java (Android version) [spacing adjusted for comparison]
<pre>         negative = true;     }      public boolean isNegative() {         return negative;     }      public boolean addPermi ssi on(Permi ssi on permi ssi on) {         if(permi ssi onSet. contains(permi ssi on)) {             return false;         } else {             permi ssi onSet. addEl ement(permi ssi on);             return true;         }     }      public boolean removePermi ssi on(Permi ssi on permi ssi on) {         return permi ssi onSet. removeEl ement(permi ssi on);     }      public boolean checkPermi ssi on(Permi ssi on permi ssi on) {         return permi ssi onSet. contains(permi ssi on);     }      public Enumeration permi ssi ons() {         return permi ssi onSet. el ements();     }      public String toString() {         StringBuffer stringbuffer = new StringBuffer();         if(negative)             stringbuffer. append("-");         else             stringbuffer. append("+");         if(user instanceof Group)             stringbuffer. append("Group. ");         else             stringbuffer. append("User. ");         stringbuffer. append((new StringBuil der()). append(user). append("="). toString());         Enumeration enumeration = permi ssi ons();         do {             if(!enumeration. hasMoreEl ements())                 break;             Permi ssi on permi ssi on = (Permi ssi on)enumeration. nextEl ement();             stringbuffer. append(permi ssi on);             if(enumeration. hasMoreEl ements())                 stringbuffer. append(", ");         } while(true);         return new String(stringbuffer);     }      public synchronized Object clone() {         Acl EntryImpl acl entryimpl = new Acl EntryImpl (user);         acl entryimpl . permi ssi onSet = (Vector)permi ssi onSet. clone();         acl entryimpl . negative = negative;         return acl entryimpl ;     } </pre>	<pre>         negative = true;     }      public boolean isNegative() {         return negative;     }      public boolean addPermi ssi on(Permi ssi on permi ssi on) {         if(permi ssi onSet. contains(permi ssi on)) {             return false;         } else {             permi ssi onSet. addEl ement(permi ssi on);             return true;         }     }      public boolean removePermi ssi on(Permi ssi on permi ssi on) {         return permi ssi onSet. removeEl ement(permi ssi on);     }      public boolean checkPermi ssi on(Permi ssi on permi ssi on) {         return permi ssi onSet. contains(permi ssi on);     }      public Enumeration permi ssi ons() {         return permi ssi onSet. el ements();     }      public String toString() {         StringBuffer stringbuffer = new StringBuffer();         if(negative)             stringbuffer. append("-");         else             stringbuffer. append("+");         if(user instanceof Group)             stringbuffer. append("Group. ");         else             stringbuffer. append("User. ");         stringbuffer. append((new StringBuil der()). append(user). append("="). toString());         Enumeration enumeration = permi ssi ons();         do {             if(!enumeration. hasMoreEl ements())                 break;             Permi ssi on permi ssi on = (Permi ssi on)enumeration. nextEl ement();             stringbuffer. append(permi ssi on);             if(enumeration. hasMoreEl ements())                 stringbuffer. append(", ");         } while(true);         return new String(stringbuffer);     }      public synchronized Object clone() {         Acl EntryImpl acl entryimpl = new Acl EntryImpl (user);         acl entryimpl . permi ssi onSet = (Vector)permi ssi onSet. clone();         acl entryimpl . negative = negative;         return acl entryimpl ;     } </pre>

AclEntryImpl.jad (decompiled version of Oracle AclEntryImpl.class) [spacing adjusted for comparison]	AclEntryImpl.java (Android version) [spacing adjusted for comparison]
<pre>public Principal getPrincipal () {     return user; }</pre>	<pre>public Principal getPrincipal () {     return user; }</pre>

## Exhibit Copyright-L

AclImpl.jad (decompiled version of Oracle AclImpl.class) [spacing adjusted for comparison]	AclImpl.java (Android version) [spacing adjusted for comparison]
<pre>// Decompiled by Jad v1.5.8g. Copyright 2001 Pavel Kouznetsov. // Jad home page: http://www.kpdus.com/jad.html // Decompiler options: fieldsfirst nonlb // Source File Name:   AclImpl.java  package sun.security.acl;  import java.security.Principal; import java.security.acl.Acl; import java.security.acl.AclEntry; import java.security.acl.Group; import java.security.acl.NotOwnerException; import java.security.acl.Permission; import java.util.Enumeration; import java.util.Hashtable; import java.util.Vector;  // Referenced classes of package sun.security.acl: //      OwnerImpl, AclEnumerator  public class AclImpl extends OwnerImpl     implements Acl {      private Hashtable allowedUsersTable;     private Hashtable allowedGroupsTable;     private Hashtable deniedUsersTable;     private Hashtable deniedGroupsTable;     private String aclName;     private Vector zeroSet;      public AclImpl(Principal principal, String s) {         super(principal);         allowedUsersTable = new Hashtable(23);         allowedGroupsTable = new Hashtable(23);         deniedUsersTable = new Hashtable(23);         deniedGroupsTable = new Hashtable(23);         aclName = null;         zeroSet = new Vector(1, 1);         try {             setName(principal, s);         }         catch(Exception exception) { }     } }</pre>	<pre>/*  * Licensed to the Apache Software Foundation (ASF) under one or more  * contributor license agreements. See the NOTICE file distributed with  * this work for additional information regarding copyright ownership.  * The ASF licenses this file to You under the Apache License, Version 2.0  * (the "License"); you may not use this file except in compliance with  * the License. You may obtain a copy of the License at  *  *      http://www.apache.org/licenses/LICENSE-2.0  *  * Unless required by applicable law or agreed to in writing, software  * distributed under the License is distributed on an "AS IS" BASIS,  * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.  * See the License for the specific language governing permissions and  * limitations under the License.  */  package org.apache.harmony.security.tests.support.acl;  import java.security.Principal; import java.security.acl.*; import java.util.*;  /**  * Additional class for verification Acl interface  */  public class AclImpl extends OwnerImpl implements Acl {      private Hashtable allowedUsersTable;     private Hashtable allowedGroupsTable;     private Hashtable deniedUsersTable;     private Hashtable deniedGroupsTable;     private String aclName;     private Vector zeroSet;      public AclImpl(Principal principal, String s) {         super(principal);         allowedUsersTable = new Hashtable(23);         allowedGroupsTable = new Hashtable(23);         deniedUsersTable = new Hashtable(23);         deniedGroupsTable = new Hashtable(23);         aclName = null;         zeroSet = new Vector(1, 1);         try {             setName(principal, s);         } catch(Exception exception) { }     } }</pre>

AclImpl.jad (decompiled version of Oracle AclImpl.class) [spacing adjusted for comparison]	AclImpl.java (Android version) [spacing adjusted for comparison]
<pre> public void setName(Principal principal, String s) throws NotOwnerException {     if(!isOwner(principal)) {         throw new NotOwnerException();     } else {         aclName = s;         return;     } }  public String getName() {     return aclName; }  public synchronized boolean addEntry(Principal principal, AclEntry aclEntry) throws NotOwnerException {     if(!isOwner(principal))         throw new NotOwnerException();     Hashtable hashtable = findTable(aclEntry);     Principal principal1 = aclEntry.getPrincipal();     if(hashtable.get(principal1) != null) {         return false;     } else {         hashtable.put(principal1, aclEntry);         return true;     } }  public synchronized boolean removeEntry(Principal principal, AclEntry aclEntry) throws NotOwnerException {     if(!isOwner(principal)) {         throw new NotOwnerException();     } else {         Hashtable hashtable = findTable(aclEntry);         Principal principal1 = aclEntry.getPrincipal();         Object obj = hashtable.remove(principal1);         return obj != null;     } }  public synchronized Enumeration getPermissions(Principal principal) {     Enumeration enumeration2 = subtract(getGroupPositive(principal), getGroupNegative(principal));     Enumeration enumeration3 = subtract(getGroupNegative(principal), getGroupPositive(principal));     Enumeration enumeration = subtract(getIndividualPositive(principal), getIndividualNegative(principal));     Enumeration enumeration1 = subtract(getIndividualNegative(principal), getIndividualPositive(principal));     Enumeration enumeration4 = subtract(enumeration2, enumeration1);     Enumeration enumeration5 = union(enumeration, enumeration4);     enumeration = subtract(getIndividualPositive(principal), getIndividualNegative(principal));     enumeration1 = subtract(getIndividualNegative(principal), getIndividualPositive(principal));     enumeration4 = subtract(enumeration3, enumeration);     Enumeration enumeration6 = union(enumeration1, enumeration4); </pre>	<pre> public void setName(Principal principal, String s) throws NotOwnerException {     if(!isOwner(principal)) {         throw new NotOwnerException();     } else {         aclName = s;         return;     } }  public String getName() {     return aclName; }  public synchronized boolean addEntry(Principal principal, AclEntry aclEntry) throws NotOwnerException {     if(!isOwner(principal)) throw new NotOwnerException();     Hashtable hashtable = findTable(aclEntry);     Principal principal1 = aclEntry.getPrincipal();     if(hashtable.get(principal1) != null) {         return false;     } else {         hashtable.put(principal1, aclEntry);         return true;     } }  public synchronized boolean removeEntry(Principal principal, AclEntry aclEntry) throws NotOwnerException {     if(!isOwner(principal)) {         throw new NotOwnerException();     } else {         Hashtable hashtable = findTable(aclEntry);         Principal principal1 = aclEntry.getPrincipal();         Object obj = hashtable.remove(principal1);         return obj != null;     } }  public synchronized Enumeration getPermissions(Principal principal) {     Enumeration enumeration2 = subtract(getGroupPositive(principal), getGroupNegative(principal));     Enumeration enumeration3 = subtract(getGroupNegative(principal), getGroupPositive(principal));     Enumeration enumeration = subtract(getIndividualPositive(principal), getIndividualNegative(principal));     Enumeration enumeration1 = subtract(getIndividualNegative(principal), getIndividualPositive(principal));     Enumeration enumeration4 = subtract(enumeration2, enumeration1);     Enumeration enumeration5 = union(enumeration, enumeration4);     enumeration = subtract(getIndividualPositive(principal), getIndividualNegative(principal));     enumeration1 = subtract(getIndividualNegative(principal), getIndividualPositive(principal));     enumeration4 = subtract(enumeration3, enumeration);     Enumeration enumeration6 = union(enumeration1, enumeration4); </pre>

AclImpl.jad (decompiled version of Oracle AclImpl.class) [spacing adjusted for comparison]	AclImpl.java (Android version) [spacing adjusted for comparison]
<pre>         return subtract(enumeration5, enumeration6);     }      public boolean checkPermission(Principal principal, Permission permission) {         for(Enumeration enumeration = getPermissions(principal); enumeration.hasMoreElements();) {             Permission permission1 = (Permission)enumeration.nextElement();             if(permission1.equals(permission))                 return true;         }         return false;     }      public synchronized Enumeration entries() {         return new AclEnumerator(this, allowedUsersTable, allowedGroupsTable, deniedUsersTable, deniedGroupsTable);     }      public String toString() {         StringBuffer stringbuffer = new StringBuffer();         for(Enumeration enumeration = entries(); enumeration.hasMoreElements(); stringbuffer.append("\n")) {             AclEntry aclentry = (AclEntry)enumeration.nextElement();             stringbuffer.append(aclentry.toString().trim());         }         return stringbuffer.toString();     }      private Hashtable findTable(AclEntry aclentry) {         Hashtable hashtable = null;         Principal principal = aclentry.getPrincipal();         if(principal instanceof Group) {             if(aclentry.isNegative())                 hashtable = deniedGroupsTable;             else                 hashtable = allowedGroupsTable;         } else             if(aclentry.isNegative())                 hashtable = deniedUsersTable;             else                 hashtable = allowedUsersTable;         return hashtable;     }      private static Enumeration union(Enumeration enumeration, Enumeration enumeration1) {         Vector vector = new Vector(20, 20);         for(; enumeration.hasMoreElements(); vector.addElement(enumeration.nextElement()));         do {             if(!enumeration1.hasMoreElements())                 break;             Object obj = enumeration1.nextElement();             if(!vector.contains(obj))                 vector.addElement(obj);         } while(true);     } </pre>	<pre>         return subtract(enumeration5, enumeration6);     }      public boolean checkPermission(Principal principal, Permission permission) {         for(Enumeration enumeration = getPermissions(principal); enumeration.hasMoreElements();) {             Permission permission1 = (Permission)enumeration.nextElement();             if(permission1.equals(permission))                 return true;         }         return false;     }      public synchronized Enumeration entries() {         return new AclEnumerator(this, allowedUsersTable, allowedGroupsTable, deniedUsersTable, deniedGroupsTable);     }      public String toString() {         StringBuffer stringbuffer = new StringBuffer();         for(Enumeration enumeration = entries(); enumeration.hasMoreElements(); stringbuffer.append("\n")) {             AclEntry aclentry = (AclEntry)enumeration.nextElement();             stringbuffer.append(aclentry.toString().trim());         }         return stringbuffer.toString();     }      private Hashtable findTable(AclEntry aclentry) {         Hashtable hashtable = null;         Principal principal = aclentry.getPrincipal();         if(principal instanceof Group) {             if(aclentry.isNegative())                 hashtable = deniedGroupsTable;             else                 hashtable = allowedGroupsTable;         } else             if(aclentry.isNegative())                 hashtable = deniedUsersTable;             else                 hashtable = allowedUsersTable;         return hashtable;     }      private static Enumeration union(Enumeration enumeration, Enumeration enumeration1) {         Vector vector = new Vector(20, 20);         for(; enumeration.hasMoreElements(); vector.addElement(enumeration.nextElement()));         do {             if(!enumeration1.hasMoreElements())                 break;             Object obj = enumeration1.nextElement();             if(!vector.contains(obj))                 vector.addElement(obj);         } while(true);     } </pre>

AclImpl.jad (decompiled version of Oracle AclImpl.class) [spacing adjusted for comparison]	AclImpl.java (Android version) [spacing adjusted for comparison]
<pre>         return vector.elements();     }      private Enumeration subtract(Enumeration enumeration, Enumeration enumeration1) {         Vector vector = new Vector(20, 20);         for(; enumeration.hasMoreElements(); vector.addElement(enumeration.nextElement()));         do {             if(!enumeration1.hasMoreElements())                 break;             Object obj = enumeration1.nextElement();             if(vector.contains(obj))                 vector.removeElement(obj);         } while(true);         return vector.elements();     }      private Enumeration getGroupPositive(Principal principal) {         Enumeration enumeration = zeroSet.elements();         Enumeration enumeration1 = allowedGroupsTable.keys();         do {             if(!enumeration1.hasMoreElements())                 break;             Group group = (Group)enumeration1.nextElement();             if(group.isMember(principal)) {                 AclEntry aclEntry = (AclEntry)allowedGroupsTable.get(group);                 enumeration = union(aclEntry.permissions(), enumeration);             }         } while(true);         return enumeration;     }      private Enumeration getGroupNegative(Principal principal) {         Enumeration enumeration = zeroSet.elements();         Enumeration enumeration1 = deniedGroupsTable.keys();         do {             if(!enumeration1.hasMoreElements())                 break;             Group group = (Group)enumeration1.nextElement();             if(group.isMember(principal)) {                 AclEntry aclEntry = (AclEntry)deniedGroupsTable.get(group);                 enumeration = union(aclEntry.permissions(), enumeration);             }         } while(true);         return enumeration;     }      private Enumeration getIndividualPositive(Principal principal) {         Enumeration enumeration = zeroSet.elements();         AclEntry aclEntry = (AclEntry)allowedUsersTable.get(principal);         if(aclEntry != null)             enumeration = aclEntry.permissions();         return enumeration;     }      private Enumeration getIndividualNegative(Principal principal) {         Enumeration enumeration = zeroSet.elements();         AclEntry aclEntry = (AclEntry)deniedUsersTable.get(principal); </pre>	<pre>         return vector.elements();     }      private Enumeration subtract(Enumeration enumeration, Enumeration enumeration1) {         Vector vector = new Vector(20, 20);         for(; enumeration.hasMoreElements(); vector.addElement(enumeration.nextElement()));         do {             if(!enumeration1.hasMoreElements())                 break;             Object obj = enumeration1.nextElement();             if(vector.contains(obj))                 vector.removeElement(obj);         } while(true);         return vector.elements();     }      private Enumeration getGroupPositive(Principal principal) {         Enumeration enumeration = zeroSet.elements();         Enumeration enumeration1 = allowedGroupsTable.keys();         do {             if(!enumeration1.hasMoreElements())                 break;             Group group = (Group)enumeration1.nextElement();             if(group.isMember(principal)) {                 AclEntry aclEntry = (AclEntry)allowedGroupsTable.get(group);                 enumeration = union(aclEntry.permissions(), enumeration);             }         } while(true);         return enumeration;     }      private Enumeration getGroupNegative(Principal principal) {         Enumeration enumeration = zeroSet.elements();         Enumeration enumeration1 = deniedGroupsTable.keys();         do {             if(!enumeration1.hasMoreElements())                 break;             Group group = (Group)enumeration1.nextElement();             if(group.isMember(principal)) {                 AclEntry aclEntry = (AclEntry)deniedGroupsTable.get(group);                 enumeration = union(aclEntry.permissions(), enumeration);             }         } while(true);         return enumeration;     }      private Enumeration getIndividualPositive(Principal principal) {         Enumeration enumeration = zeroSet.elements();         AclEntry aclEntry = (AclEntry)allowedUsersTable.get(principal);         if(aclEntry != null)             enumeration = aclEntry.permissions();         return enumeration;     }      private Enumeration getIndividualNegative(Principal principal) {         Enumeration enumeration = zeroSet.elements();         AclEntry aclEntry = (AclEntry)deniedUsersTable.get(principal); </pre>

AclImpl.jad (decompiled version of Oracle AclImpl.class) [spacing adjusted for comparison]	AclImpl.java (Android version) [spacing adjusted for comparison]
<pre>if(aclentry != null)     enumeration = aclentry.permissions(); return enumeration; }</pre>	<pre>if(aclentry != null)     enumeration = aclentry.permissions(); return enumeration; }</pre>



**Exhibit Copyright-M**

GroupImpl.jad (decompiled version of Oralce GroupImpl.class) [spacing adjusted for comparison]	GroupImpl.java (Android version) [spacing adjusted for comparison]
<pre>// Decompiled by Jad v1.5.8g. Copyright 2001 Pavel Kouznetsov. // Jad home page: http://www.kpdus.com/jad.html // Decompiler options: fieldsfirst nonlb // Source File Name:   GroupImpl.java  package sun.security.acl;  import java.security.Principal; import java.security.acl.Group; import java.util.Enumeration; import java.util.Vector;  public class GroupImpl     implements Group {      private Vector groupMembers;     private String group;      public GroupImpl(String s) {         groupMembers = new Vector(50, 100);         group = s;     }      public boolean addMember(Principal principal) {         if(groupMembers.contains(principal))             return false;         if(group.equals(principal.toString())) {             throw new IllegalArgumentException();         } else {             groupMembers.addElement(principal);             return true;         }     }      public boolean removeMember(Principal principal) {         return groupMembers.removeElement(principal);     }      public Enumeration members() {         return groupMembers.elements();     } }</pre>	<pre>/*  * Licensed to the Apache Software Foundation (ASF) under one or more  * contributor license agreements. See the NOTICE file distributed with  * this work for additional information regarding copyright ownership.  * The ASF licenses this file to You under the Apache License, Version 2.0  * (the "License"); you may not use this file except in compliance with  * the License. You may obtain a copy of the License at  *  *     http://www.apache.org/licenses/LICENSE-2.0  *  * Unless required by applicable law or agreed to in writing, software  * distributed under the License is distributed on an "AS IS" BASIS,  * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.  * See the License for the specific language governing permissions and  * limitations under the License.  */  package org.apache.harmony.security.tests.support.acl;  import java.security.Principal; import java.security.acl.Group; import java.util.Enumeration; import java.util.Vector;  /**  * Additional class for verification Group interface  */ public class GroupImpl implements Group {      private Vector groupMembers;     private String group;      public GroupImpl(String s) {         groupMembers = new Vector(50, 100);         group = s;     }      public boolean addMember(Principal principal) {         if(groupMembers.contains(principal))             return false;         if(group.equals(principal.toString())) {             throw new IllegalArgumentException();         } else {             groupMembers.addElement(principal);             return true;         }     }      public boolean removeMember(Principal principal) {         return groupMembers.removeElement(principal);     }      public Enumeration members() {         return groupMembers.elements();     } }</pre>

GroupImpl.jad (decompiled version of Oralce GroupImpl.class) [spacing adjusted for comparison]	GroupImpl.java (Android version) [spacing adjusted for comparison]
<pre> public boolean equals(Object obj) {     if(this == obj)         return true;     if(! (obj instanceof Group)) {         return false;     } else {         Group group1 = (Group)obj;         return group.equals(group1.toString());     } }  public boolean equals(Group group1) {     return equals(group1); }  public String toString() {     return group; }  public int hashCode() {     return group.hashCode(); }  public boolean isMember(Principal principal) {     if(groupMembers.contains(principal)) {         return true;     } else {         Vector vector = new Vector(10);         return isMemberRecurse(principal, vector);     } }  public String getName() {     return group; }  boolean isMemberRecurse(Principal principal, Vector vector) {     for(Enumeration enumeration = members(); enumeration.hasMoreElements();) {         boolean flag = false;         Principal principal1 = (Principal)enumeration.nextElement();         if(principal1.equals(principal))             return true;         if(principal1 instanceof GroupImpl) {             GroupImpl groupimpl = (GroupImpl)principal1;             vector.addElement(this);             if(!vector.contains(groupimpl))                 flag = groupimpl.isMemberRecurse(principal, vector);         } else             if(principal1 instanceof Group) {                 Group group1 = (Group)principal1;                 if(!vector.contains(group1))                     flag = group1.isMember(principal);             }         if(flag)             return flag;     }     return false; } </pre>	<pre> public boolean equals(Object obj) {     if(this == obj)         return true;     if(! (obj instanceof Group)) {         return false;     } else {         Group group1 = (Group)obj;         return group.equals(group1.toString());     } }  public boolean equals(Group group1) {     return equals(group1); }  public String toString() {     return group; }  public int hashCode() {     return group.hashCode(); }  public boolean isMember(Principal principal) {     if(groupMembers.contains(principal)) {         return true;     } else {         Vector vector = new Vector(10);         return isMemberRecurse(principal, vector);     } }  public String getName() {     return group; }  boolean isMemberRecurse(Principal principal, Vector vector) {     for(Enumeration enumeration = members(); enumeration.hasMoreElements();) {         boolean flag = false;         Principal principal1 = (Principal)enumeration.nextElement();         if(principal1.equals(principal))             return true;         if(principal1 instanceof GroupImpl) {             GroupImpl groupimpl = (GroupImpl)principal1;             vector.addElement(this);             if(!vector.contains(groupimpl))                 flag = groupimpl.isMemberRecurse(principal, vector);         } else if(principal1 instanceof Group) {             Group group1 = (Group)principal1;             if(!vector.contains(group1)) flag = group1.isMember(principal);         }         if(flag) return flag;     }     return false; } </pre>

GroupImpl.jad (decompiled version of Oralce GroupImpl.class) [spacing adjusted for comparison]	GroupImpl.java (Android version) [spacing adjusted for comparison]
<pre>} }</pre>	

**Exhibit Copyright-N**

OwnerImpl.jad (decompiled version of Oracle OwnerImpl.class) [spacing adjusted for comparison]	OwnerImpl.java (Android version) [spacing adjusted for comparison]
<pre>// Decompiled by Jad v1.5.8g. Copyright 2001 Pavel Kouznetsov. // Jad home page: http://www.kpdus.com/jad.html // Decompiler options: fieldsfirst nonlb // Source File Name:   OwnerImpl.java  package sun.security.acl;  import java.security.Principal; import java.security.acl.Group; import java.security.acl.LastOwnerException; import java.security.acl.NotOwnerException; import java.security.acl.Owner; import java.util.Enumeration;  // Referenced classes of package sun.security.acl: //      GroupImpl  public class OwnerImpl     implements Owner {      private Group ownerGroup;      public OwnerImpl(Principal principal) {         ownerGroup = new GroupImpl("AclOwners");         ownerGroup.addMember(principal);     }      public synchronized boolean addOwner(Principal principal, Principal principal1) throws NotOwnerException {          if(!isOwner(principal)) {             throw new NotOwnerException();         } else {             ownerGroup.addMember(principal1);             return false;         }     } }</pre>	<pre>/*  * Licensed to the Apache Software Foundation (ASF) under one or more  * contributor license agreements. See the NOTICE file distributed with  * this work for additional information regarding copyright ownership.  * The ASF licenses this file to You under the Apache License, Version 2.0  * (the "License"); you may not use this file except in compliance with  * the License. You may obtain a copy of the License at  *  *      http://www.apache.org/licenses/LICENSE-2.0  *  * Unless required by applicable law or agreed to in writing, software  * distributed under the License is distributed on an "AS IS" BASIS,  * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.  * See the License for the specific language governing permissions and  * limitations under the License.  */  package org.apache.harmony.security.tests.support.acl;  import java.security.Principal; import java.security.acl.*; import java.util.Enumeration;  /**  * Additional class for verification Owner interface  */ public class OwnerImpl implements Owner {      private Group ownerGroup;      public OwnerImpl(Principal principal) {         ownerGroup = new GroupImpl("AclOwners");         ownerGroup.addMember(principal);     }      public synchronized boolean addOwner(Principal principal, Principal principal1)         throws NotOwnerException {          if(!isOwner(principal))         {             throw new NotOwnerException();         } else {             if (ownerGroup.isMember(principal1)) return false;             if (!ownerGroup.isMember(principal1)) {                 ownerGroup.addMember(principal1);                 return true;             }         }         return false;     } }</pre>

OwnerImpl.jad (decompiled version of Oracle OwnerImpl.class) [spacing adjusted for comparison]	OwnerImpl.java (Android version) [spacing adjusted for comparison]
<pre> public synchronized boolean deleteOwner(Principal principal, Principal principal1) throws NotOwnerException, LastOwnerException {     if(!isOwner(principal))         throw new NotOwnerException();      Enumeration enumeration = ownerGroup.members();     Object obj = enumeration.nextElement();     if(enumeration.hasMoreElements())         return ownerGroup.removeMember(principal1);     else         throw new LastOwnerException(); }  public synchronized boolean isOwner(Principal principal) {     return ownerGroup.isMember(principal); } </pre>	<pre> public synchronized boolean deleteOwner(Principal principal, Principal principal1)     throws NotOwnerException, LastOwnerException {     if(!isOwner(principal)) throw new NotOwnerException();     Enumeration enumeration = ownerGroup.members();     Object obj = enumeration.nextElement();     if(enumeration.hasMoreElements()) {         return ownerGroup.removeMember(principal1);     } else {         throw new LastOwnerException();     } }  public synchronized boolean isOwner(Principal principal) {     return ownerGroup.isMember(principal); } </pre>

**Exhibit Copyright-O**

PermissionImpl.jad (decompiled version of Oracle PermissionImpl.class) [spacing adjusted for comparison]	PermissionImpl.java (Android version) [spacing adjusted for comparison]
<pre>// Decompiled by Jad v1.5.8g. Copyright 2001 Pavel Kouznetsov. // Jad home page: http://www.kpdus.com/jad.html // Decompiler options: fieldsfirst nonlb // Source File Name:    PermissionImpl.java  package sun.security.acl;  import java.security.acl.Permission;  public class PermissionImpl     implements Permission {      private String permission;      public PermissionImpl(String s) {         permission = s;     }      public boolean equals(Object obj) {         if(obj instanceof Permission) {             Permission permission1 = (Permission)obj;             return permission.equals(permission1.toString());         } else {             return false;         }     }      public String toString() {         return permission;     }      public int hashCode() {         return toString().hashCode();     } }</pre>	<pre>/*  * Licensed to the Apache Software Foundation (ASF) under one or more  * contributor license agreements. See the NOTICE file distributed with  * this work for additional information regarding copyright ownership.  * The ASF licenses this file to You under the Apache License, Version 2.0  * (the "License"); you may not use this file except in compliance with  * the License. You may obtain a copy of the License at  *  *      http://www.apache.org/licenses/LICENSE-2.0  *  * Unless required by applicable law or agreed to in writing, software  * distributed under the License is distributed on an "AS IS" BASIS,  * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.  * See the License for the specific language governing permissions and  * limitations under the License.  */  package org.apache.harmony.security.tests.support.acl;  import java.security.acl.Permission;  /**  * Additional class for verification Permission interface  */ public class PermissionImpl implements Permission {      private String permission;      public PermissionImpl(String s) {         permission = s;     }      public boolean equals(Object obj) {         if(obj instanceof Permission) {             Permission permission1 = (Permission)obj;             return permission.equals(permission1.toString());         } else {             return false;         }     }      public String toString() {         return permission;     }      /*      * public int hashCode() {      *     return toString().hashCode();      * }*/ }</pre>

**Exhibit Copyright-P**

PrincipalImpl.jad (decompiled version of Oracle PrincipalImpl.class) [spacing adjusted for comparison]	PrincipalImpl.java (Android version) [spacing adjusted for comparison]
<pre>// Decompiled by Jad v1.5.8g. Copyright 2001 Pavel Kouznetsov. // Jad home page: http://www.kpdus.com/jad.html // Decompiler options: fieldsfirst nonlb // Source File Name:    PrincipalImpl.java  package sun.security.acl;  import java.security.Principal;  public class PrincipalImpl     implements Principal {      private String user;      public PrincipalImpl(String s) {         user = s;     }      public boolean equals(Object obj) {         if(obj instanceof PrincipalImpl) {             PrincipalImpl principalimpl = (PrincipalImpl)obj;             return user.equals(principalimpl.toString());         } else {             return false;         }     }      public String toString() {         return user;     }      public int hashCode() {         return user.hashCode();     }      public String getName() {         return user;     } }</pre>	<pre>/*  * Licensed to the Apache Software Foundation (ASF) under one or more  * contributor license agreements. See the NOTICE file distributed with  * this work for additional information regarding copyright ownership.  * The ASF licenses this file to You under the Apache License, Version 2.0  * (the "License"); you may not use this file except in compliance with  * the License. You may obtain a copy of the License at  *  *      http://www.apache.org/licenses/LICENSE-2.0  *  * Unless required by applicable law or agreed to in writing, software  * distributed under the License is distributed on an "AS IS" BASIS,  * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.  * See the License for the specific language governing permissions and  * limitations under the License.  */  package org.apache.harmony.security.tests.support.acl;  import java.security.Principal;  /**  * Additional class for verification Principal interface  */ public class PrincipalImpl implements Principal {      private String user;      public PrincipalImpl(String s) {         user = s;     }      public boolean equals(Object obj) {         if(obj instanceof PrincipalImpl) {             PrincipalImpl principalimpl = (PrincipalImpl)obj;             return user.equals(principalimpl.toString());         } else {             return false;         }     }      public String toString() {         return user;     }      public int hashCode() {         return user.hashCode();     }      public String getName() {         return user;     } }</pre>

**Exhibit Copyright-Q**

AclEnumerator.jad (decompiled version of Oracle AclEnumerator.class) [spacing adjusted for comparison]	AclEnumerator.java (Android version) [spacing adjusted for comparison]
<pre>// Decompiled by Jad v1.5.8g. Copyright 2001 Pavel Kouznetsov. // Jad home page: http://www.kpdus.com/jad.html // Decompiler options: packimports(3) fieldsfirst nonlb // Source File Name:  AclImpl.java  package sun.security.acl;  import java.security.acl.Acl; import java.util.*;  final class AclEnumerator     implements Enumeration {      Acl acl;     Enumeration u1;     Enumeration u2;     Enumeration g1;     Enumeration g2;      AclEnumerator(Acl acl1, Hashtable hashtable, Hashtable hashtable1,         Hashtable hashtable2, Hashtable hashtable3) {         acl = acl1;         u1 = hashtable.elements();         u2 = hashtable2.elements();         g1 = hashtable1.elements();         g2 = hashtable3.elements();     }      public boolean hasMoreElements() {         return u1.hasMoreElements()    u2.hasMoreElements()    g1.hasMoreElements()    g2.hasMoreElements();     }      public Object nextElement() {         Acl acl1 = acl;         JVM INSTR monitorenter ;         if(u1.hasMoreElements())             return u1.nextElement();         if(!u2.hasMoreElements()) goto _L2; else goto _L1 _L1:         u2.nextElement();         acl1;         JVM INSTR monitorexit ;         return;</pre>	<pre>/*  * Licensed to the Apache Software Foundation (ASF) under one or more  * contributor license agreements. See the NOTICE file distributed with  * this work for additional information regarding copyright ownership.  * The ASF licenses this file to You under the Apache License, Version 2.0  * (the "License"); you may not use this file except in compliance with  * the License. You may obtain a copy of the License at  *  *      http://www.apache.org/licenses/LICENSE-2.0  *  * Unless required by applicable law or agreed to in writing, software  * distributed under the License is distributed on an "AS IS" BASIS,  * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.  * See the License for the specific language governing permissions and  * limitations under the License.  */  package org.apache.harmony.security.tests.support.acl;  import java.security.acl.Acl; import java.util.*;  final class AclEnumerator implements Enumeration {      Acl acl;     Enumeration u1;     Enumeration u2;     Enumeration g1;     Enumeration g2;      AclEnumerator(Acl acl1, Hashtable hashtable, Hashtable hashtable1,         Hashtable hashtable2, Hashtable hashtable3) {         acl = acl1;         u1 = hashtable.elements();         u2 = hashtable2.elements();         g1 = hashtable1.elements();         g2 = hashtable3.elements();     }      public boolean hasMoreElements() {         return u1.hasMoreElements()    u2.hasMoreElements()    g1.hasMoreElements()    g2.hasMoreElements();     }      public Object nextElement() {         Acl acl1 = acl;         if(u2.hasMoreElements()) return u2.nextElement();         if(g1.hasMoreElements()) return g1.nextElement();         if(u1.hasMoreElements()) return u1.nextElement();         if(g2.hasMoreElements()) return g2.nextElement();         return acl1;     } }</pre>



AclEnumerator.jad (decompiled version of Oracle AclEnumerator.class) [spacing adjusted for comparison]	AclEnumerator.java (Android version) [spacing adjusted for comparison]
<pre> _L2: _L3:   if(!g1.hasMoreElements()) goto _L4; else goto _L3       g1.nextElement();       acl1;       JVM INSTR moni torex i t ;       return; _L4: _L5:   if(!g2.hasMoreElements()) goto _L6; else goto _L5       g2.nextElement();       acl1;       JVM INSTR moni torex i t ;       return; _L6:       acl1;       JVM INSTR moni torex i t ;       goto _L7       Excepti on excepti on;       excepti on;       throw excepti on; _L7:       throw new NoSuchElementExcepti on("Acl Enumerator");     }   } </pre>	

**Exhibit Copyright-R**

<p>CodeSource.java (Java version)</p> <p>[spacing adjusted for comparison]</p>	<p>CodeSourceTest.java (Android version)</p> <p>[spacing adjusted for comparison]</p>
<p><u>Lines 242-59</u></p> <pre> * &lt;li&gt; If this object's port (getLocation().getPort()) is not * equal to -1 (that is, if a port is specified), it must equal * &lt;i&gt;codesource&lt;/i&gt;'s port. * * &lt;li&gt; If this object's file (getLocation().getFile()) doesn't equal * &lt;i&gt;codesource&lt;/i&gt;'s file, then the following checks are made: * If this object's file ends with "/-", * then &lt;i&gt;codesource&lt;/i&gt;'s file must start with this object's * file (exclusive the trailing "-"). * If this object's file ends with a "/*", * then &lt;i&gt;codesource&lt;/i&gt;'s file must start with this object's * file and must not have any further "/" separators. * If this object's file doesn't end with a "/", * then &lt;i&gt;codesource&lt;/i&gt;'s file must match this object's * file with a '/' appended. * * &lt;li&gt; If this object's reference (getLocation().getRef()) is * not null, it must equal &lt;i&gt;codesource&lt;/i&gt;'s reference. </pre>	<pre> <u>Lines 598-601</u> /** * If this object's port (getLocation().getPort()) is not equal to -1 (that * is, if a port is specified), it must equal codesource's port. */  <u>Lines 629-32</u> /** * If this object's file (getLocation().getFile()) doesn't equal * codesource's file, then the following checks are made: ... */  <u>Lines 645-48</u> /** * ... If this object's file ends with "/-", then codesource's file must * start with this object's file (exclusive the trailing "-"). */  <u>Lines 667-81</u> /** * ... If this object's file ends with a "/*", then codesource's file must * start with this object's file and must not have any further "/" * separators. */  <u>Lines 693-96</u> /** * ... If this object's file doesn't end with a "/", then codesource's file * must match this object's file with a '/' appended. */  <u>Lines 711-14</u> /** * If this object's reference (getLocation().getRef()) is not null, it must * equal codesource's reference. */ </pre>

## Exhibit Copyright-S

Excperts from CollectionCertStoreParameters.java (Java version)	Exceprts from CollectionCertStoreParameters.java (Android version)
<p><u>Lines 43-68</u></p> <pre> /**  * Creates an instance of &lt;code&gt;CollectionCertStoreParameters&lt;/code&gt;  * which will allow certificates and CRLs to be retrieved from the  * specified &lt;code&gt;Collection&lt;/code&gt;. If the specified  * &lt;code&gt;Collection&lt;/code&gt; contains an object that is not a  * &lt;code&gt;Certificate&lt;/code&gt; or &lt;code&gt;CRL&lt;/code&gt;, that object will be  * ignored by the Collection &lt;code&gt;CertStore&lt;/code&gt;.  * &lt;p&gt;  * The &lt;code&gt;Collection&lt;/code&gt; is &lt;b&gt;not&lt;/b&gt; copied. Instead, a  * reference is used. This allows the caller to subsequently add or  * remove &lt;code&gt;Certificates&lt;/code&gt; or &lt;code&gt;CRL&lt;/code&gt;s from the  * &lt;code&gt;Collection&lt;/code&gt;, thus changing the set of  * &lt;code&gt;Certificates&lt;/code&gt; or &lt;code&gt;CRL&lt;/code&gt;s available to the  * Collection &lt;code&gt;CertStore&lt;/code&gt;. The Collection &lt;code&gt;CertStore&lt;/code&gt;  * will not modify the contents of the &lt;code&gt;Collection&lt;/code&gt;.  * &lt;p&gt;  * If the &lt;code&gt;Collection&lt;/code&gt; will be modified by one thread while  * another thread is calling a method of a Collection &lt;code&gt;CertStore&lt;/code&gt;  * that has been initialized with this &lt;code&gt;Collection&lt;/code&gt;, the  * &lt;code&gt;Collection&lt;/code&gt; must have fail-fast iterators.  *  * @param collection a &lt;code&gt;Collection&lt;/code&gt; of  * &lt;code&gt;Certificate&lt;/code&gt;s and &lt;code&gt;CRL&lt;/code&gt;s  * @exception NullPointerException if &lt;code&gt;collection&lt;/code&gt; is  * &lt;code&gt;null&lt;/code&gt;  */ </pre>	<p><u>Lines 110-116</u></p> <pre> /**  * Test #2 for &lt;code&gt;CollectionCertStoreParameters(Collection)&lt;/code&gt;  * constructor&lt;br&gt;  * Assertion: If the specified &lt;code&gt;Collection&lt;/code&gt; contains an object  * that is not a &lt;code&gt;Certificate&lt;/code&gt; or &lt;code&gt;CRL&lt;/code&gt;, that object  * will be ignored by the Collection &lt;code&gt;CertStore&lt;/code&gt;.  */ </pre> <p><u>Lines 132-140</u></p> <pre> /**  * Test #3 for &lt;code&gt;CollectionCertStoreParameters(Collection)&lt;/code&gt;  * constructor&lt;br&gt;  * Assertion: The Collection is not copied. Instead, a reference is used.  * This allows the caller to subsequently add or remove Certificates or  * CRLs from the Collection, thus changing the set of Certificates or CRLs  * available to the Collection CertStore. The Collection CertStore will  * not modify the contents of the Collection  */ </pre>
<p><u>Lines 75-79</u></p> <pre> /**  * Creates an instance of &lt;code&gt;CollectionCertStoreParameters&lt;/code&gt; with  * the default parameter values (an empty and immutable  * &lt;code&gt;Collection&lt;/code&gt;).  */ </pre>	<p><u>Lines 50-54</u></p> <pre> /**  * Test #1 for &lt;code&gt;CollectionCertStoreParameters()&lt;/code&gt; constructor&lt;br&gt;  * Assertion: Creates an instance of CollectionCertStoreParameters  * with the default parameter values (an empty and immutable Collection)  */ </pre>